Flutter-Workshop

Julian Hartl

Overview

Why Flutter?

Flutter Basics

Dart Crash-Course

Developing a Shopping List app

Building your own app

Outlook

Why Flutter?

What even is Flutter?

What even is Flutter?

"Flutter is an open source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase."

- Open source UI software development toolkit
- Supports many platforms
- Single codebase

Why Flutter?

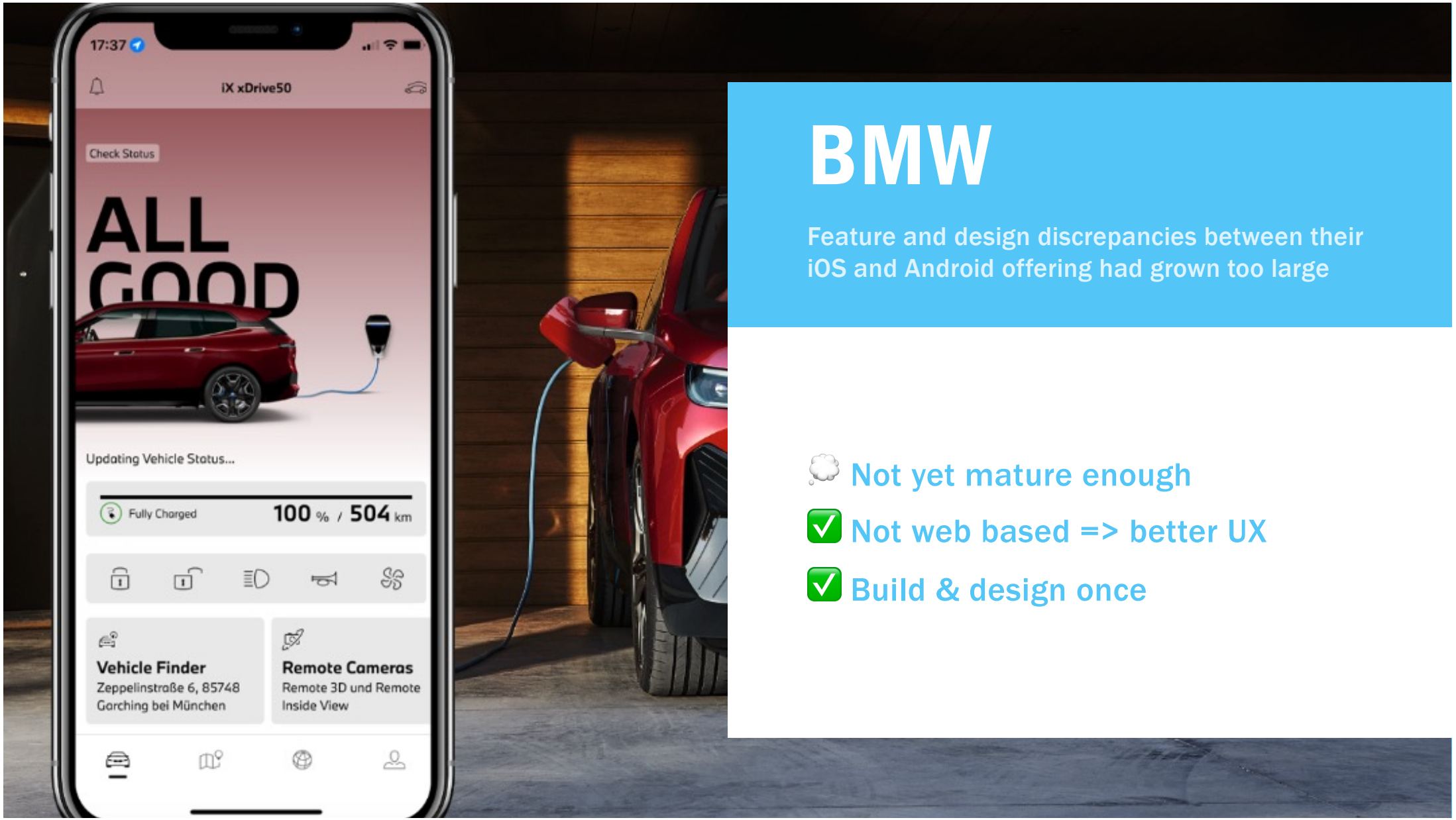
What does Flutter offer?

What does Flutter offer?

- Cross-Platform development (Android, iOS, Web, Windows, macOS, Linux, Fuchsia)
- Performant due to Ahead-of-Time (AOT) Compilation
- Expressive and flexible UI (large set of prebuilt components)
- Hot reload (instant UI updates)
- Consistent behavior across all platforms
- Strong community and ecosystem (wide adoption in the industry)

Why Flutter?

Applications in the real world



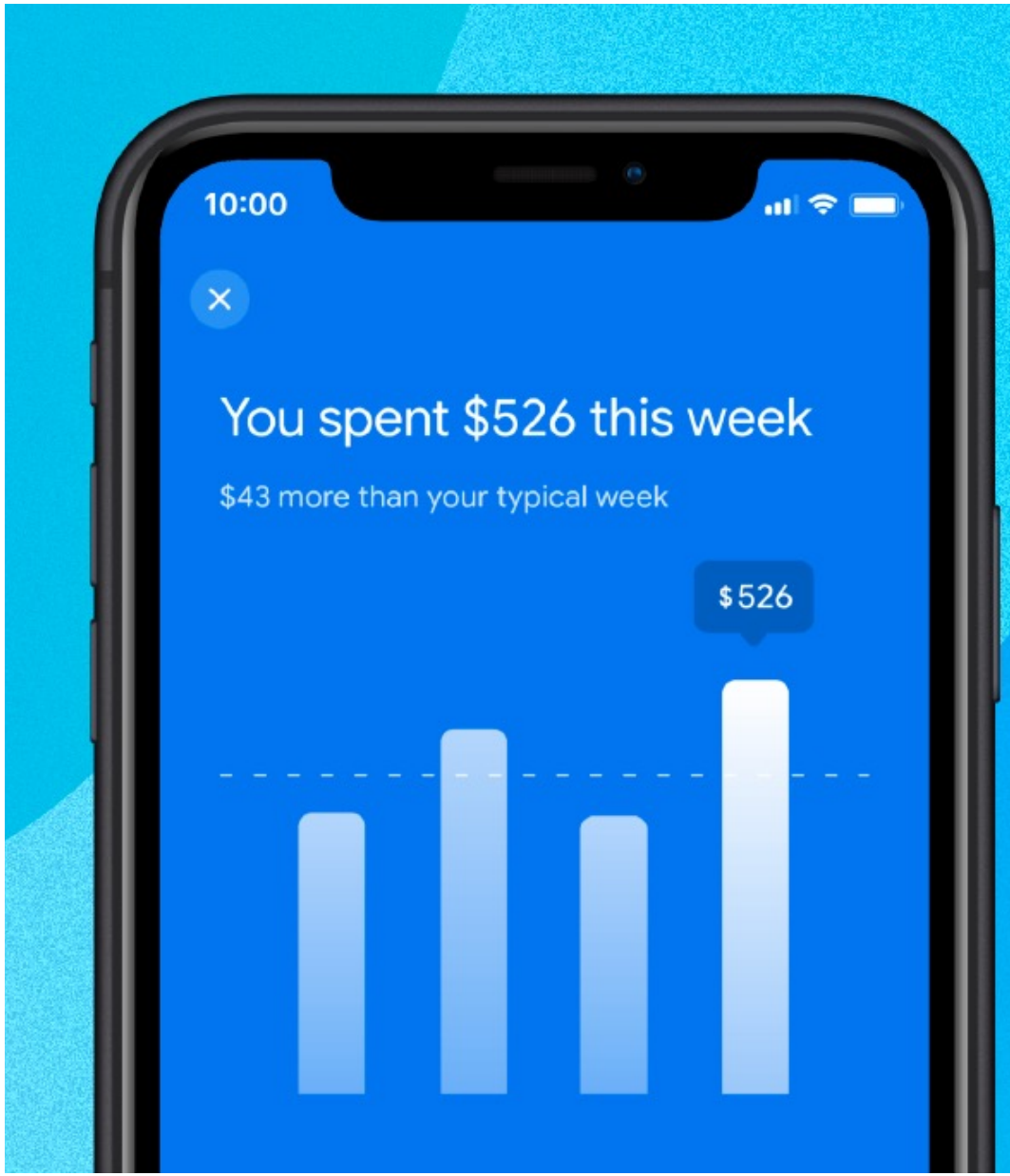
BMW

Feature and design discrepancies between their iOS and Android offering had grown too large

💭 Not yet mature enough

✅ Not web based => better UX

✅ Build & design once



Google-Pay

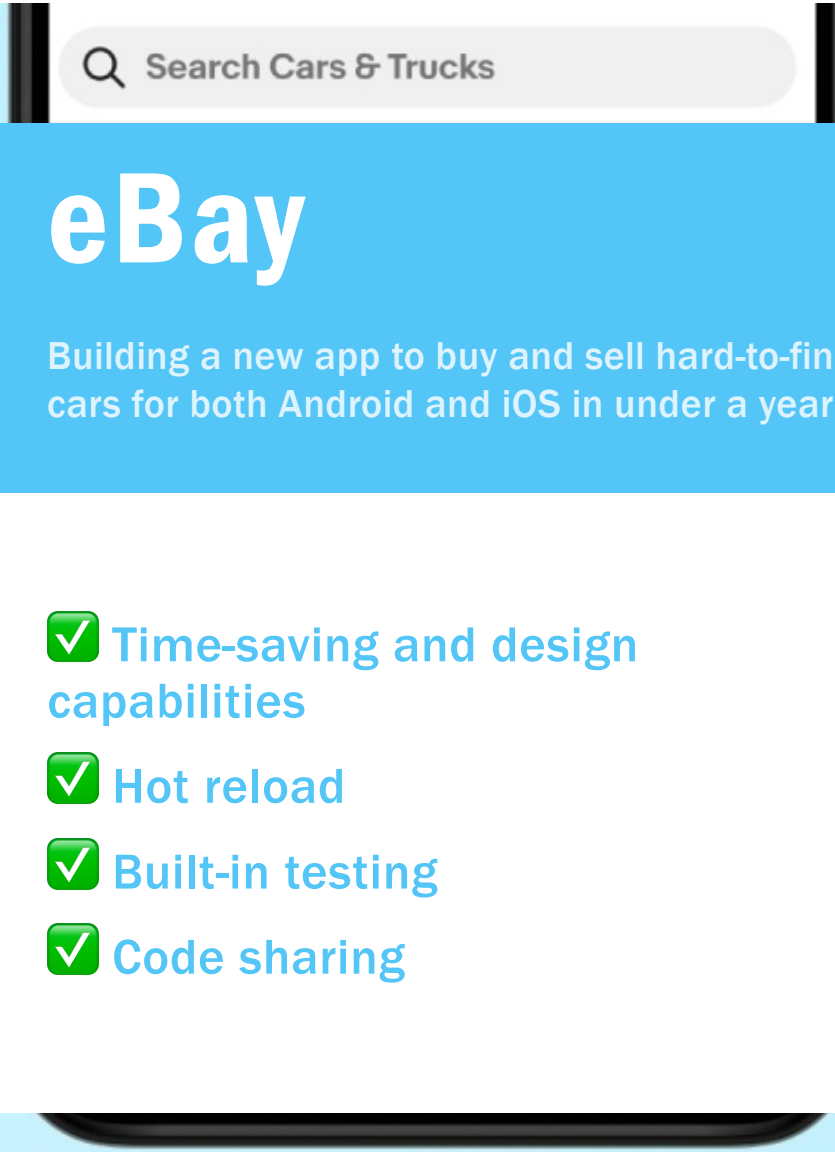
Building every feature twice

💭 Heavy upfront investment of engineering resources (retraining, reapproval, ...)

✅ 70% reduction in engineering effort

✅ 35% reduction in lines of code

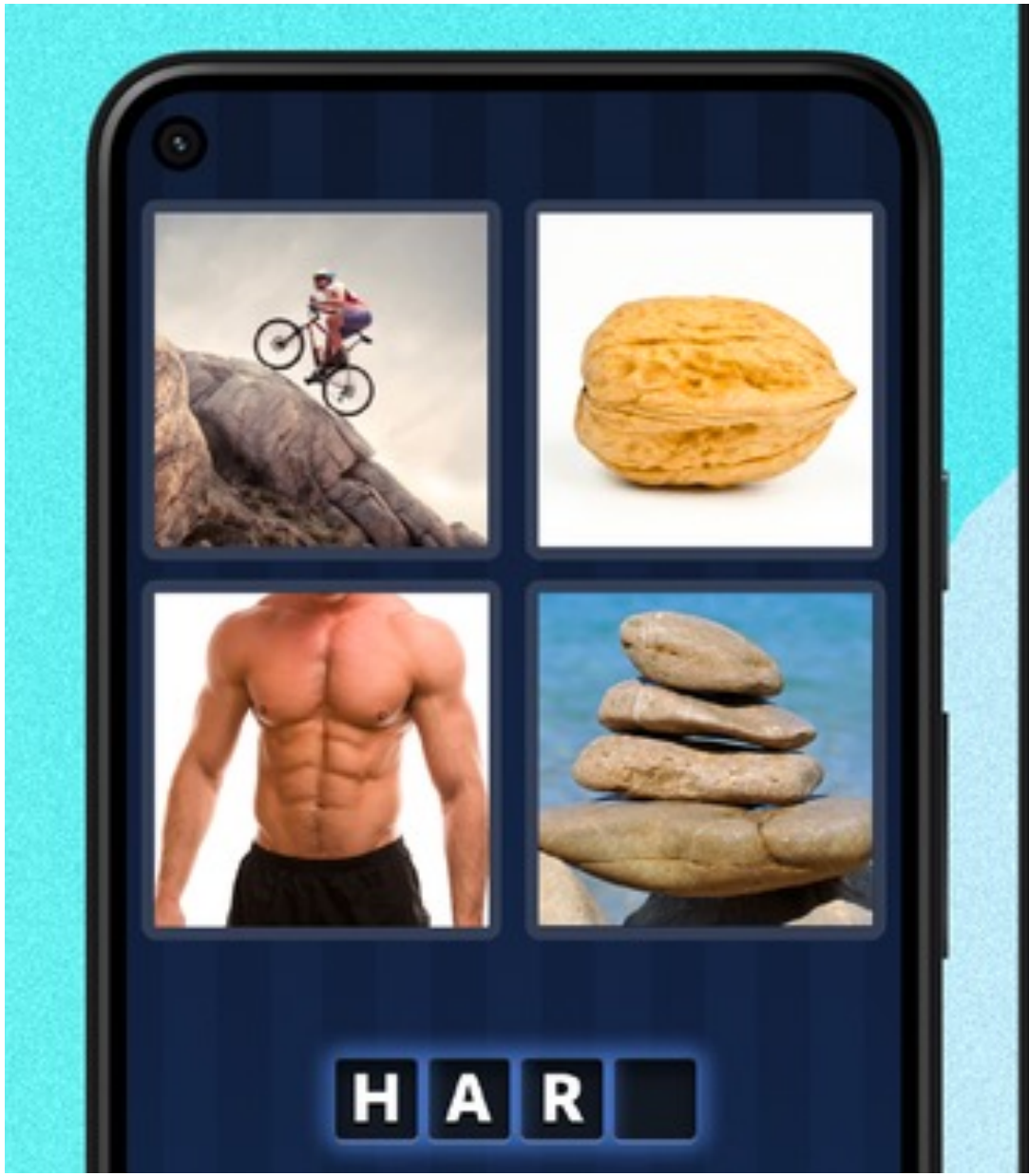
✅ Saved about 60-70% of their engineer's time



eBay

Building a new app to buy and sell hard-to-find cars for both Android and iOS in under a year

- ✅ Time-saving and design capabilities
- ✅ Hot reload
- ✅ Built-in testing
- ✅ Code sharing



4 Pics 1 Word

Giving 4 Pics 1 Word a tuneup

- ✅ No more troubleshooting cross-platform functionality
- ✅ Performs offline on low-end devices
- ✅ 99.9% crash-free rate

Why Flutter?

Flutter compared to React Native

Flutter vs. React Native

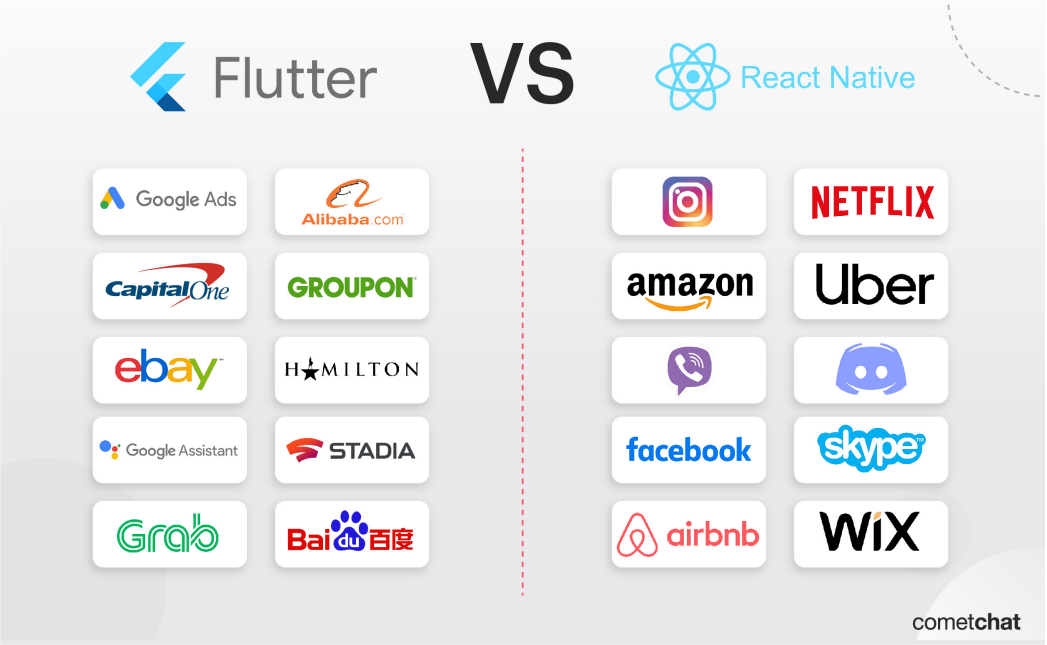
Flutter

- Comprehensive set of prebuilt UI components following Material Design & Cupertino guidelines
- AOT compiled to native code, ensuring faster startup and performance
- Has a powerful graphics engine (Skia) that supports complex customizations and graphics (animations, ...)
- Although younger, it has a rapidly growing community

React Native

- Core components, but for more complete UI, developers often rely on third-party libraries
- Uses JavaScript, which is interpreted at runtime and therefore only close to native performance
- While possible, achieving the same level of complexity as Flutter might require additional work & third-party libraries
- Established community with a vast number of libraries

Flutter vs. React Native



Why Flutter?

Flutter compared to native development

Flutter vs. Native

Flutter

- Cross-Platform
- Consistent design
- One codebase
- May require additional work for platform-specific features

Native

- Supports only one platform
- Design can vary between devices
- Two separate codebases
- Direct access to native functionalities

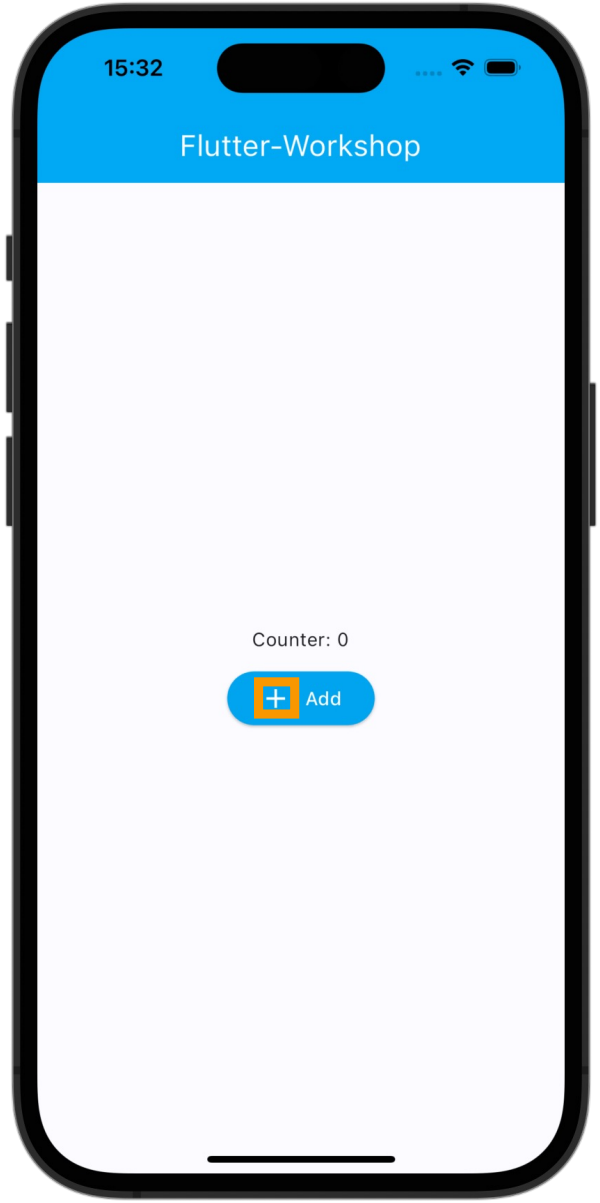
Summary

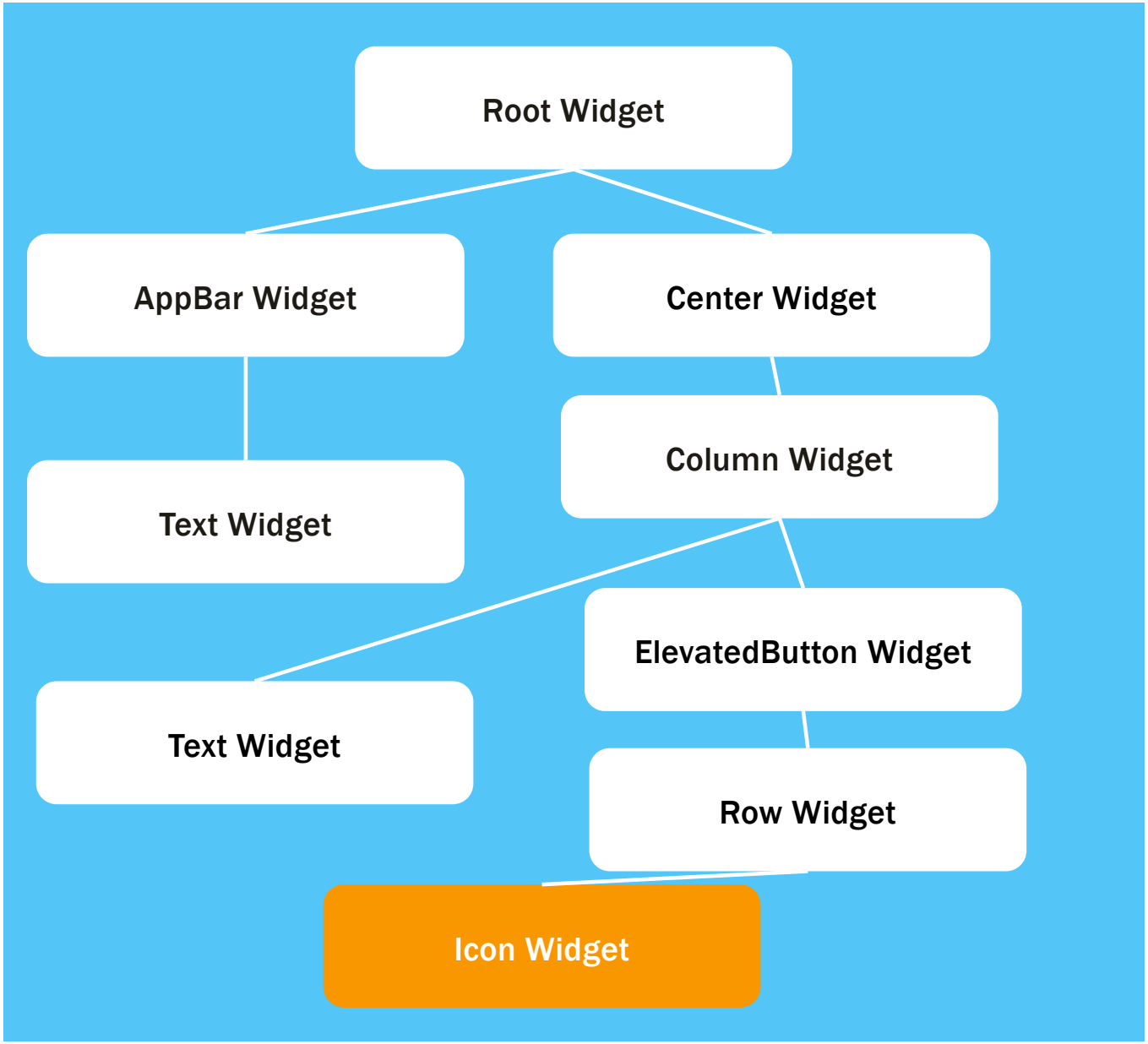
- Build once for all platforms
- Native performance
- Rich ecosystem & wide adoption
- Consistent design due to Skia Rendering-Engine

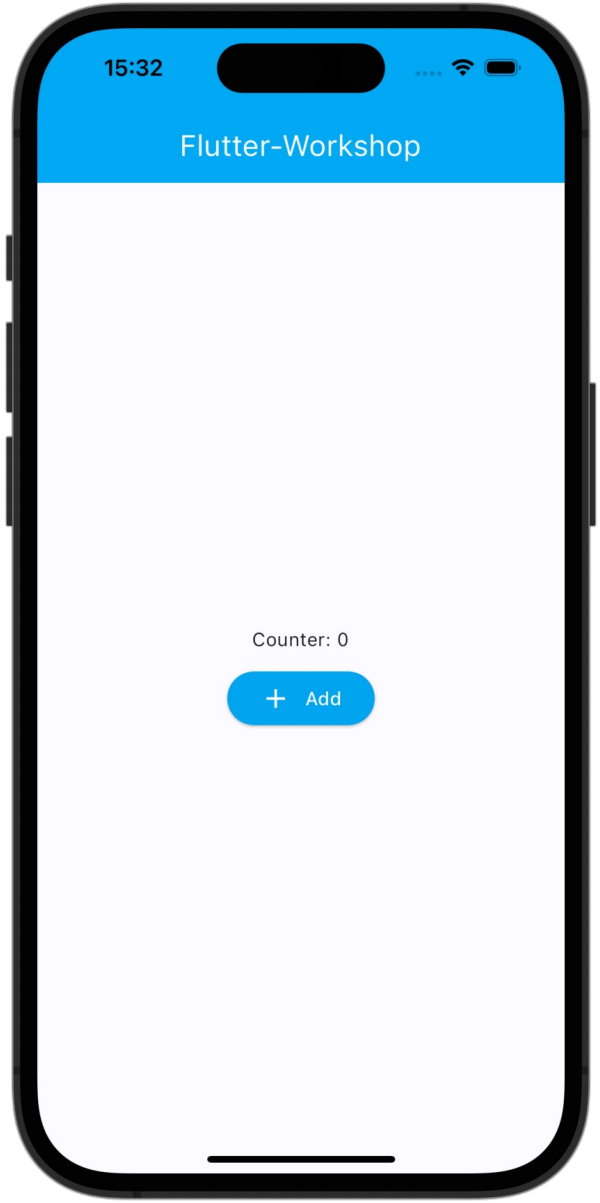
Flutter Basics

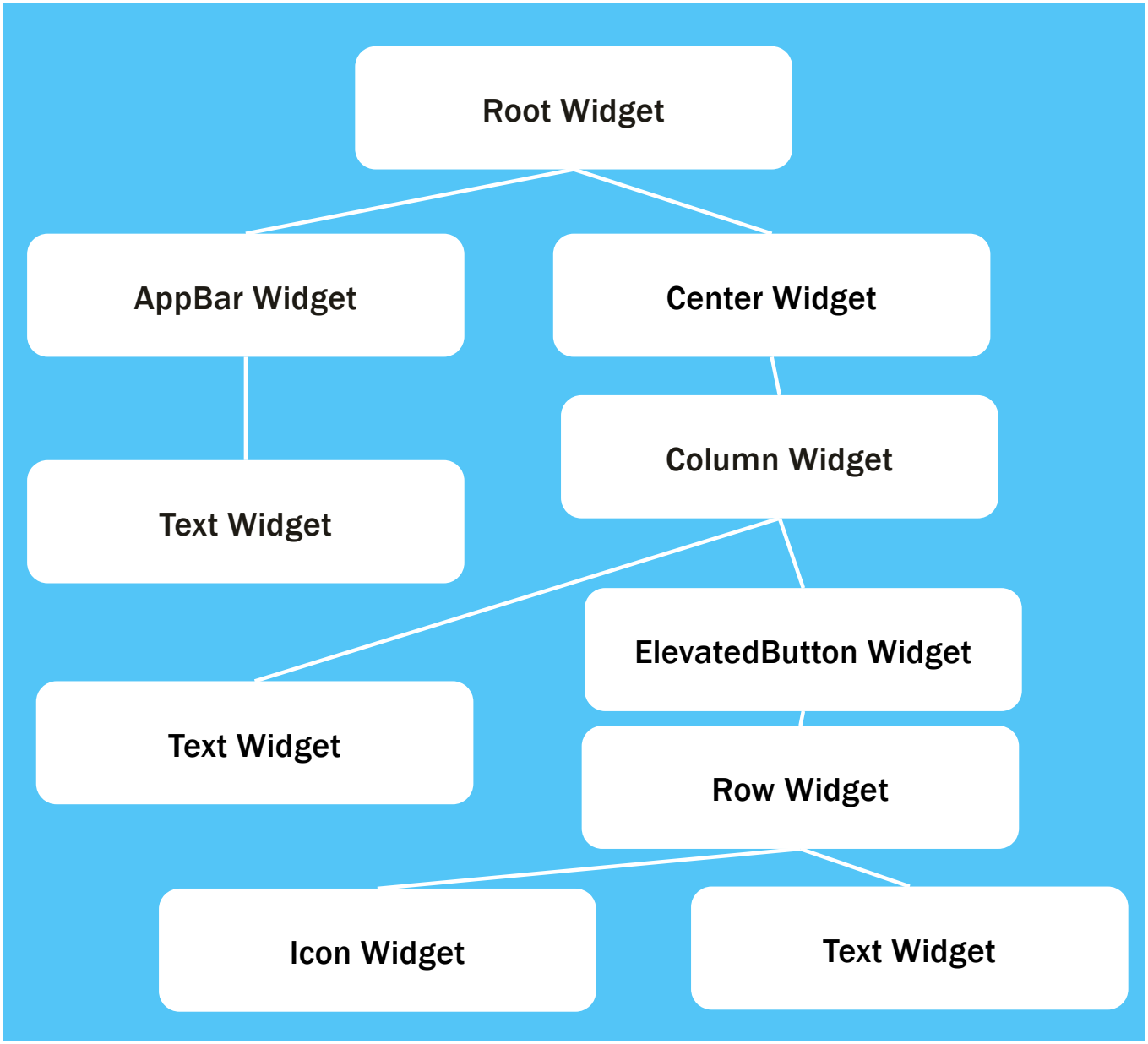
Widgets


Every-
thing
is a
Widget


15:32
Flutter-Workshop
Counter: 0
Add


Root Widget
AppBar Widget
Center Widget
Text Widget
Column Widget
ElevatedButton Widget
Text Widget
Row Widget
Icon Widget


15:32
Flutter-Workshop
Counter: 0
Add


Root Widget
AppBar Widget
Center Widget
Text Widget
Column Widget
ElevatedButton Widget
Text Widget
Row Widget
Icon Widget
Text Widget

Widgets

Properties

Properties

Text Widget	ElevatedButton Widget	Column Widget	AppBar Widget
style	style	mainAxisAlignment	title
textAlign	onPressed	crossAxisAlignment	backgroundColor
overflow	...	...	leading
maxLines			...
...			

Widgets

Stateless vs. Stateful

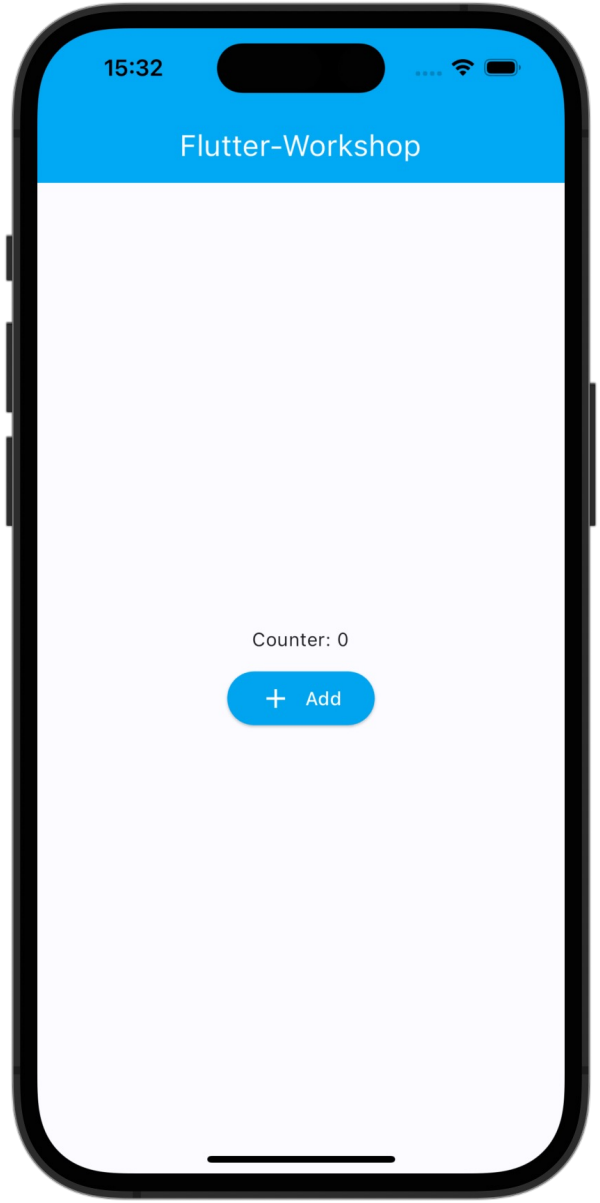
Stateless vs. Stateful

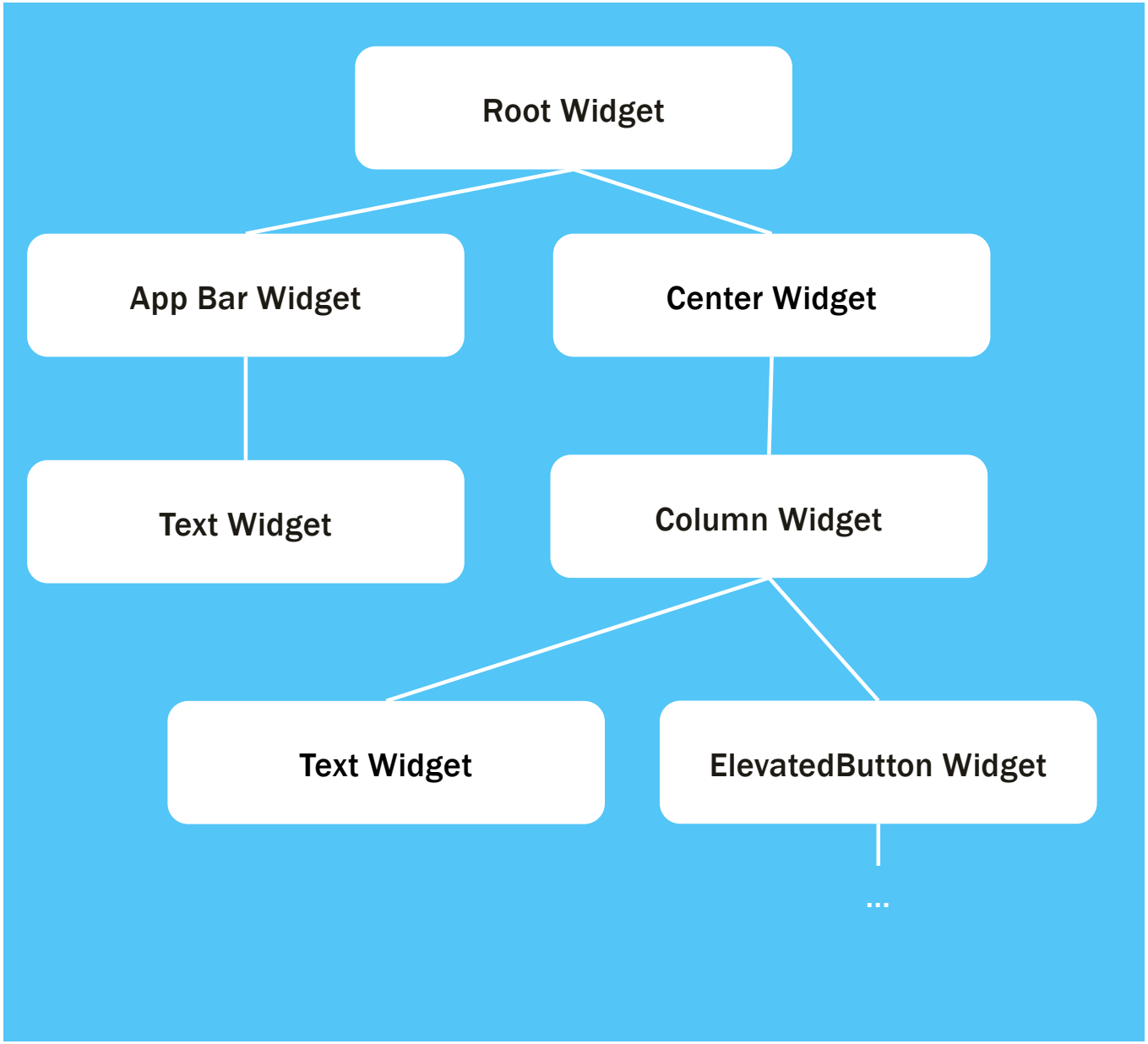
Stateless

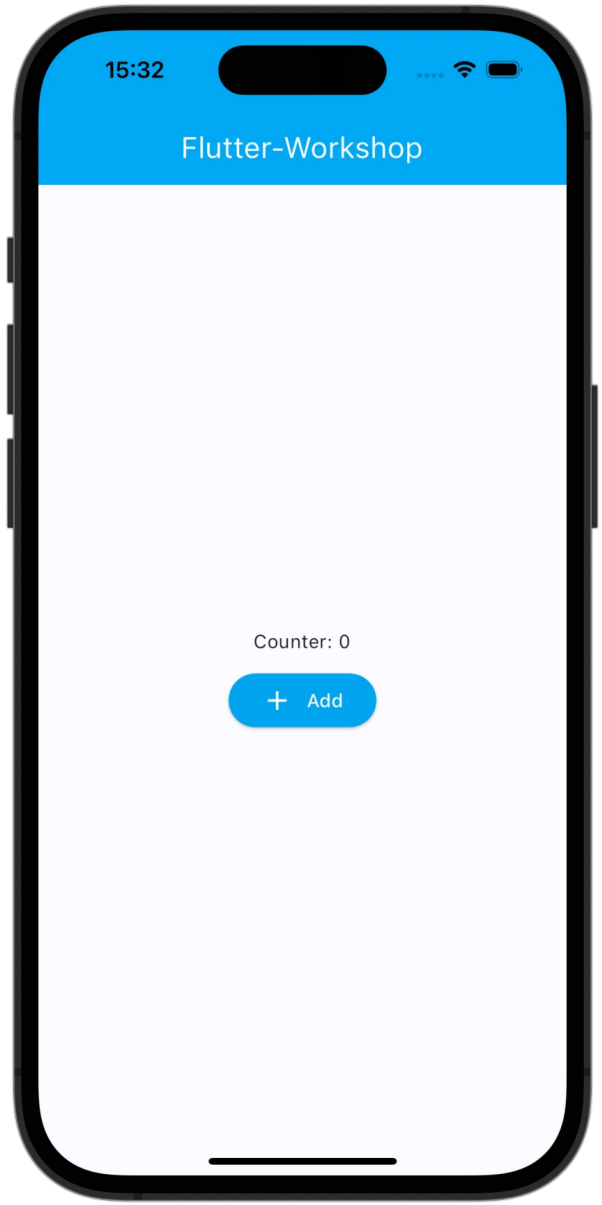
- “A widget that does not require mutable state.“
- Only needs to rerender when state in an ancestor widget changes

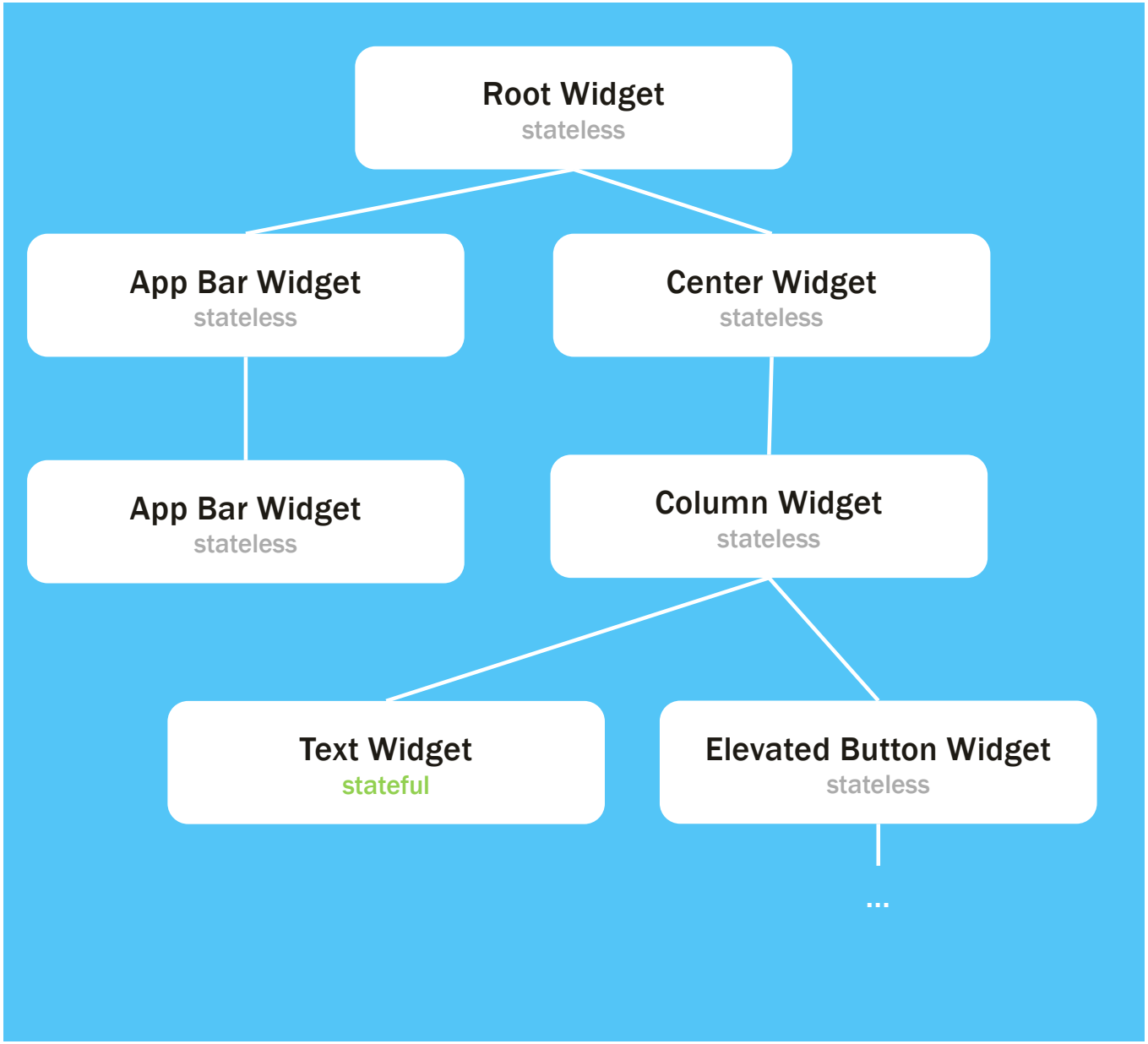
Stateful

- “A widget that has mutable state.“
- Rerenders itself and all children every time its state changes


15:32
Flutter-Workshop
Counter: 0
Add


Root Widget
App Bar Widget
Center Widget
Text Widget
Column Widget
Text Widget
ElevatedButton Widget
...


15:32
Flutter-Workshop
Counter: 0
Add

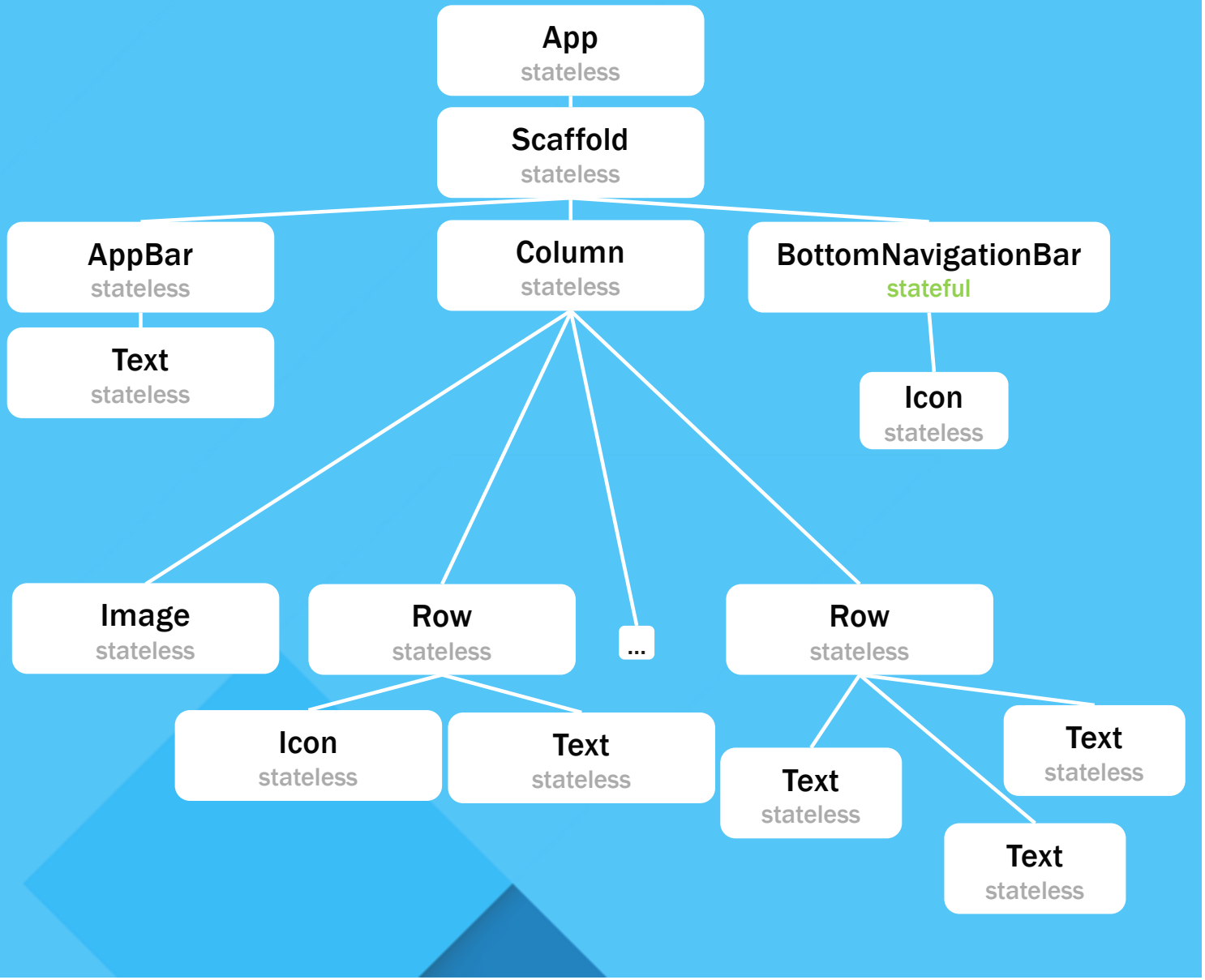

Root Widget
stateless
App Bar Widget
stateless
Center Widget
stateless
App Bar Widget
stateless
Column Widget
stateless
Text Widget
stateful
Elevated Button Widget
stateless
...

Flutter Basics

Now it’s your turn to find the Widgets


#UNSschreibtmanWIR
SPIELVEREINIGUNG
UNTERHACHING
• 1925 •
TICKETSHOP
FANSHOP
UNSERE TEAMS
HACHING SCHAUT HIN
HACHING FUSSBALLSCHULE
EINSTELLUNGEN
DATENSCHUTZ
IMPRESSUM
KONTAKT


#UNSschreibtmanWIR
SPIELVEREINIGUNG
UNTERHACHING
1925
TICKETSHOP
FANSHOP
UNSERE TEAMS
HACHING SCHAUT HIN
HACHING FUSSBALLSCHULE
EINSTELLUNGEN
DATENSCHUTZ
IMPRESSUM
KONTAKT


App
stateless
Scaffold
stateless
AppBar
stateless
Column
stateless
BottomNavigationBar
stateful
Text
stateless
Icon
stateless
Image
stateless
Row
stateless
...
Row
stateless
Icon
stateless
Text
stateless
Text
stateless
Text
stateless
Text
stateless

Flutter Basics

What is BuildContext?

BuildContext

- Every widget has a unique BuildContext
- A reference or handle to the location of a widget within the widget tree
- Allows you to access data or functions from ancestor widgets

Summary

- ”Everything is a widget”
- A page is just a tree of widgets (including itself)
- There are stateless & stateful widgets
- Use BuildContext to navigate the tree

Dart Crash-Course

Built-in types

```dart
int integer = 42;

String string = 'This is a string';

bool boolean = true || false;

double d = 3.1415;

List<int> numbers = [1, 2, 3, 4, 5];
```

Further built-in types

- Record/Tuple
- Set
- Map
- Null
- Object (Everything except for Null)
- Void

Null safety

```dart
int? number = null;
```

👆 Nullable type

Control flow (conditional branching)

```
if (condition) {
  // do something
} else if (anotherCondition) {
  // do something else
} else {
  // do something else
}
```

Control flow (while loop)

```
while (!isDone()) {
  doSomething();
}
```

Functions

```dart
void doSomething() {
  print('Hello World!');
}

bool isDone() {
  return true;
}

while (!isDone()) {
  doSomething();
}
```

Functions

```
bool isDone(int number) {
  return number == 3;
}

int number = 0;
while(!isDone(number)) {
  doSomething();
  number++;
}
```

Anonymous functions

```dart
void doSomething(
  void Function(int count) something,
  int count,
) {
  something(count);
}

doSomething((count) {
  print('Hello World! $count');
}, number);
```

👈 Functions are also objects

Control flow (for loop)

```
var message = StringBuffer('Dart is fun');
for (var i = 0; i < 5; i++) {
  message.write('!');
}
```

Control flow (enhanced for loop)

```dart
final participants = <Participant>[];
for (final participant in participants) {
  print('Hallo ${participant.name}!');
}
```

Classes

```dart
class Participant {
  final String name;

  const Participant(this.name);
}
```

👆 All members can be known at compile time

Classes

```dart
class Participant {
  final String name;
  final int age;
  final String role;
  String _password;
  Participant(
    this.age, {
    required this.name,
    this.role = 'member',
    required String password,
  }) : _password = password;

  void greet() {
    print('Hello $name!');
  }
  bool get hasFun => true;
  set password(String newPassword) {
    if (newPassword.length < 8) {
      throw Exception('Password must be at least 8 characters long');
    }
    _password = newPassword;
  }
}
```

Classes

```dart
final participant = Participant(
  19,
  name: 'Julian Hartl',
  password: 'mysecretpassword',
);
participant.greet();
particpant.password = 'myverysecretpassword';
print(particpant.hasFun);
```

Positional parameter 👉

👈 Named parameters

👈 Call the greet method

Set new password 👉

👈 Use hasFun getter

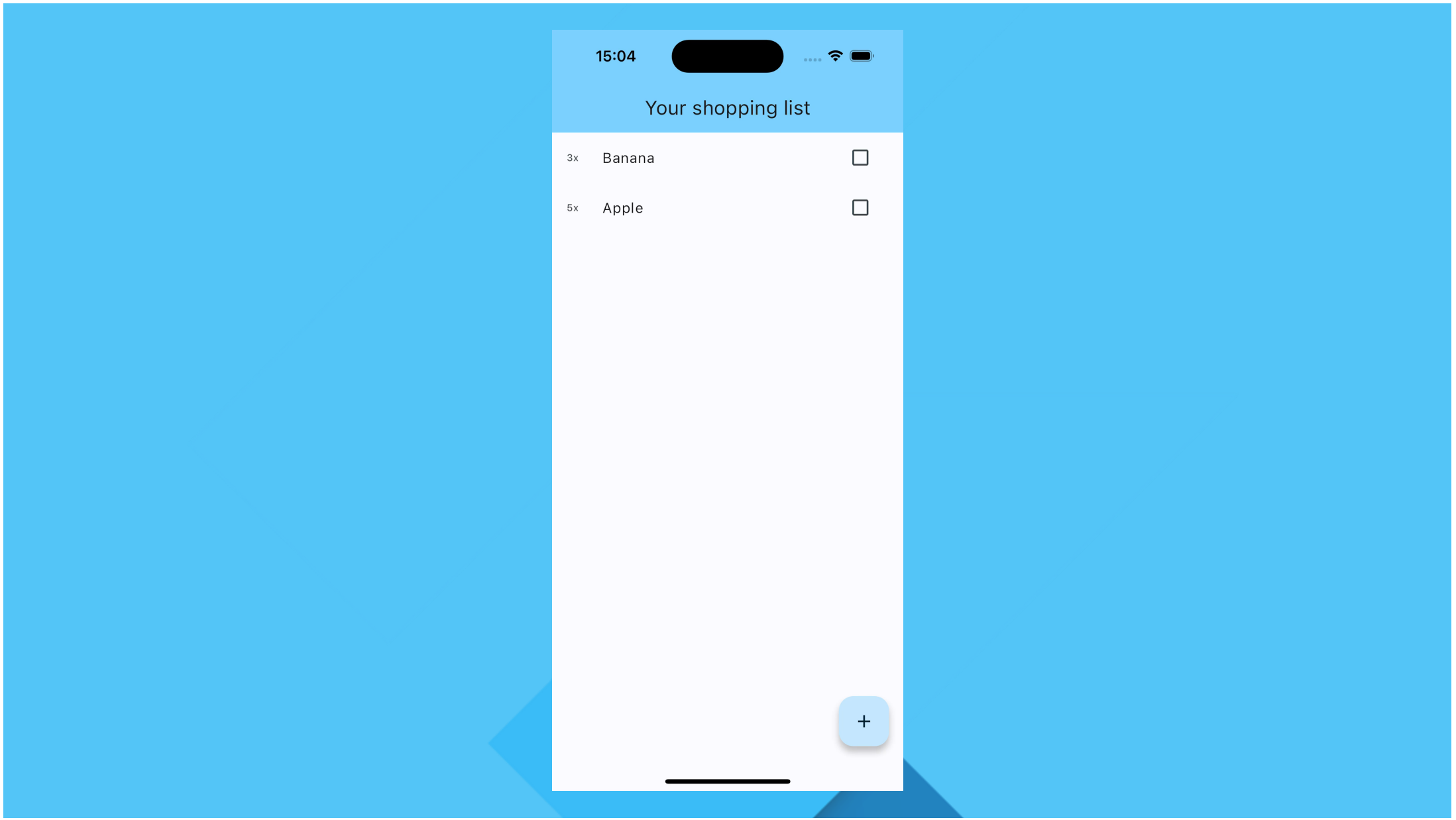
Summary

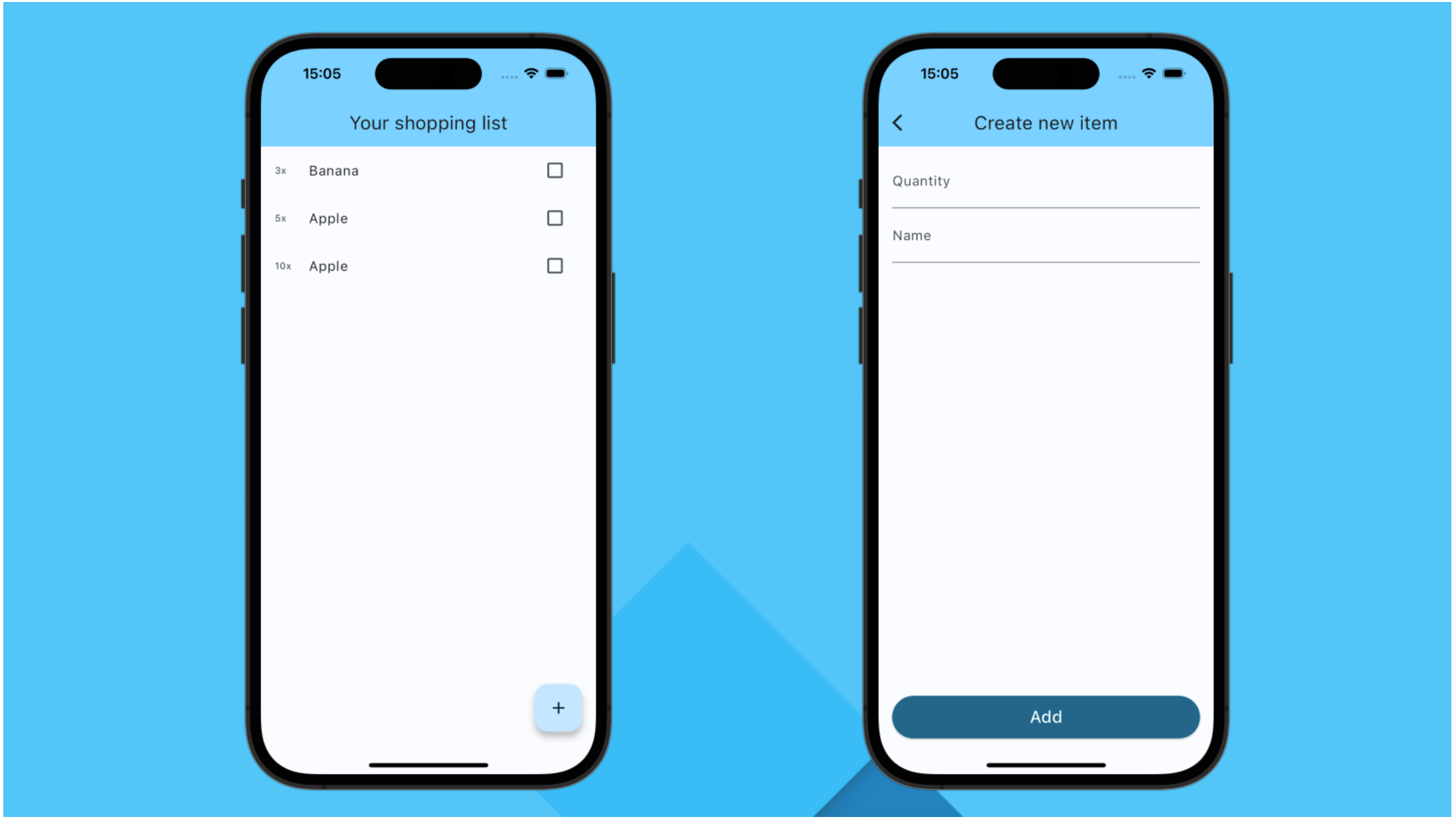
- Very Java/JavaScript like language
- Easy to pick up
- Lots of modern features

Shopping List App

Demo

Shopping List App

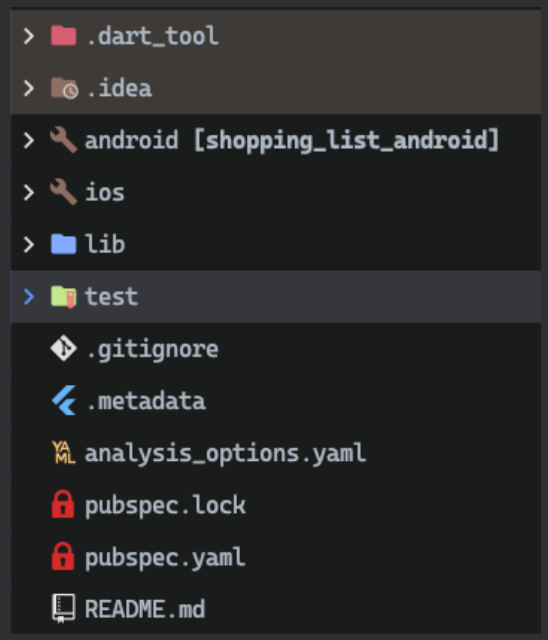

15:04
Your shopping list
3x
Banana
5x
Apple
+


15:05
Your shopping list
3x Banana
5x Apple
10x Apple
+
15:05
Create new item
Quantity
Name
Add

“Live”-Coding

Shopping List App

```
flutter create shopping_list --org "dev.julianhartl" -t app --platforms android,ios -a kotlin -i swift
```



👈 “The *.dart_tool* directory, which is new in Dart 2, is used by pub and other tools.“

👈 Code specific to the operating systems

👈 Actual Flutter related code

👈 Here go your tests 😜

👈 Linter configuration

👈 Dependencies lock file (similar to yarn.lock or package-json.lock)

👈 Dependencies, assets, project configuration

pubspec.yaml

Project configuration

```yaml
name: shopping_list
description: A new Flutter project.
publish_to: 'none'

version: 1.0.0+1

environment:
  sdk: '>=3.1.3 <4.0.0'

dependencies:
  flutter:
    sdk: flutter
  cupertino_icons: ^1.0.2

dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_lints: ^2.0.0
```

👈 Project configuration

👈 Version + build number

👈 Supported dart versions

👈 Any packages which are included in the bundled app

👈 Any packages that are needed during development, but should not be bundled in the final app

```yaml
flutter:
  uses-material-design: true
  assets:
    - images/a_dot_burr.jpeg
    - images/a_dot_ham.jpeg
  fonts:
    - family: Schyler
      fonts:
        - asset: fonts/Schyler-Regular.ttf
        - asset: fonts/Schyler-Italic.ttf
          style: italic
    - family: Trajan Pro
      fonts:
        - asset: fonts/TrajanPro.ttf
        - asset: fonts/TrajanPro_Bold.ttf
          weight: 700
```

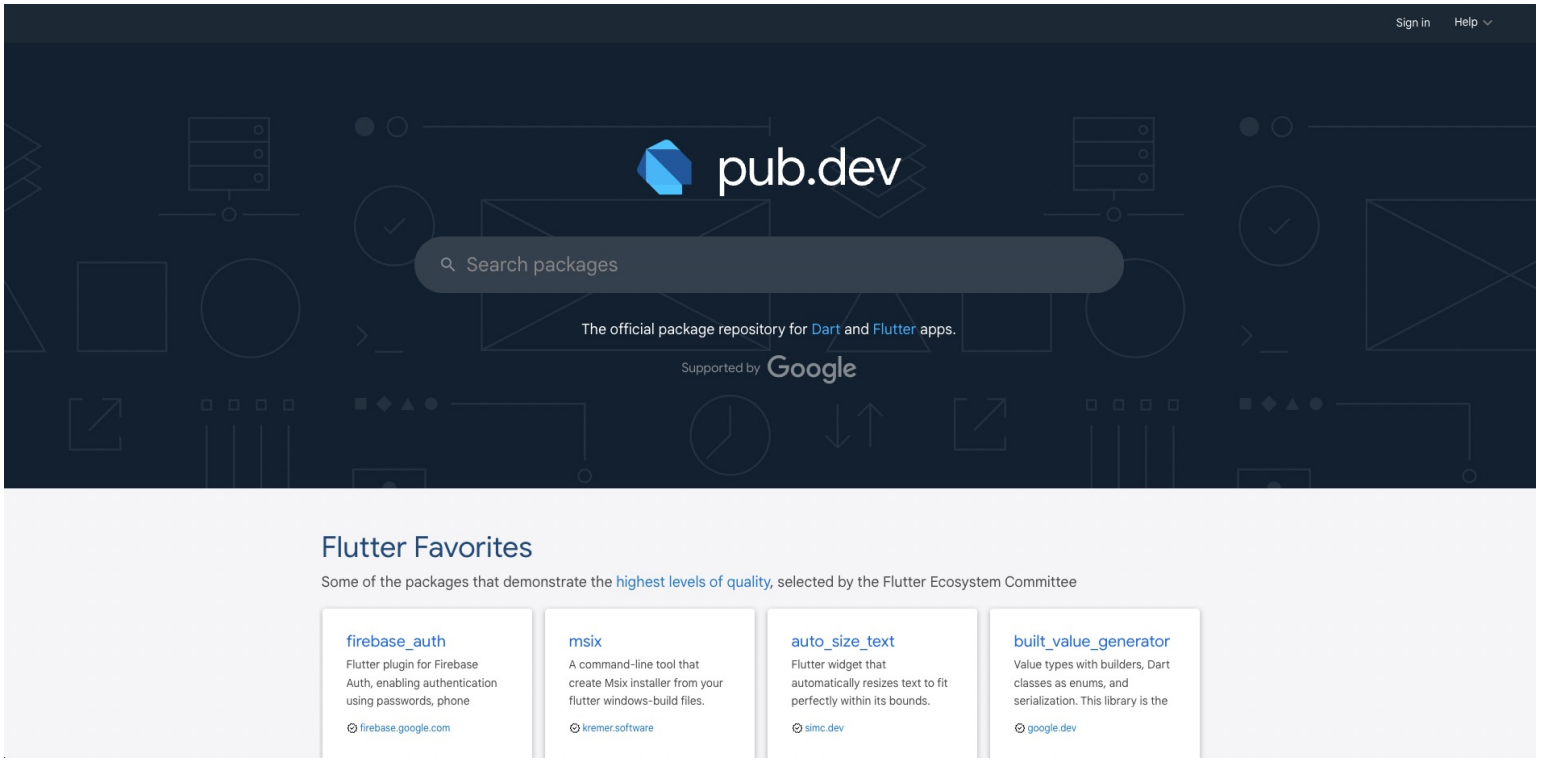
👈 Any assets, that should be included in the bundled app

👈 Definition of customs fonts

Pub

The package manager

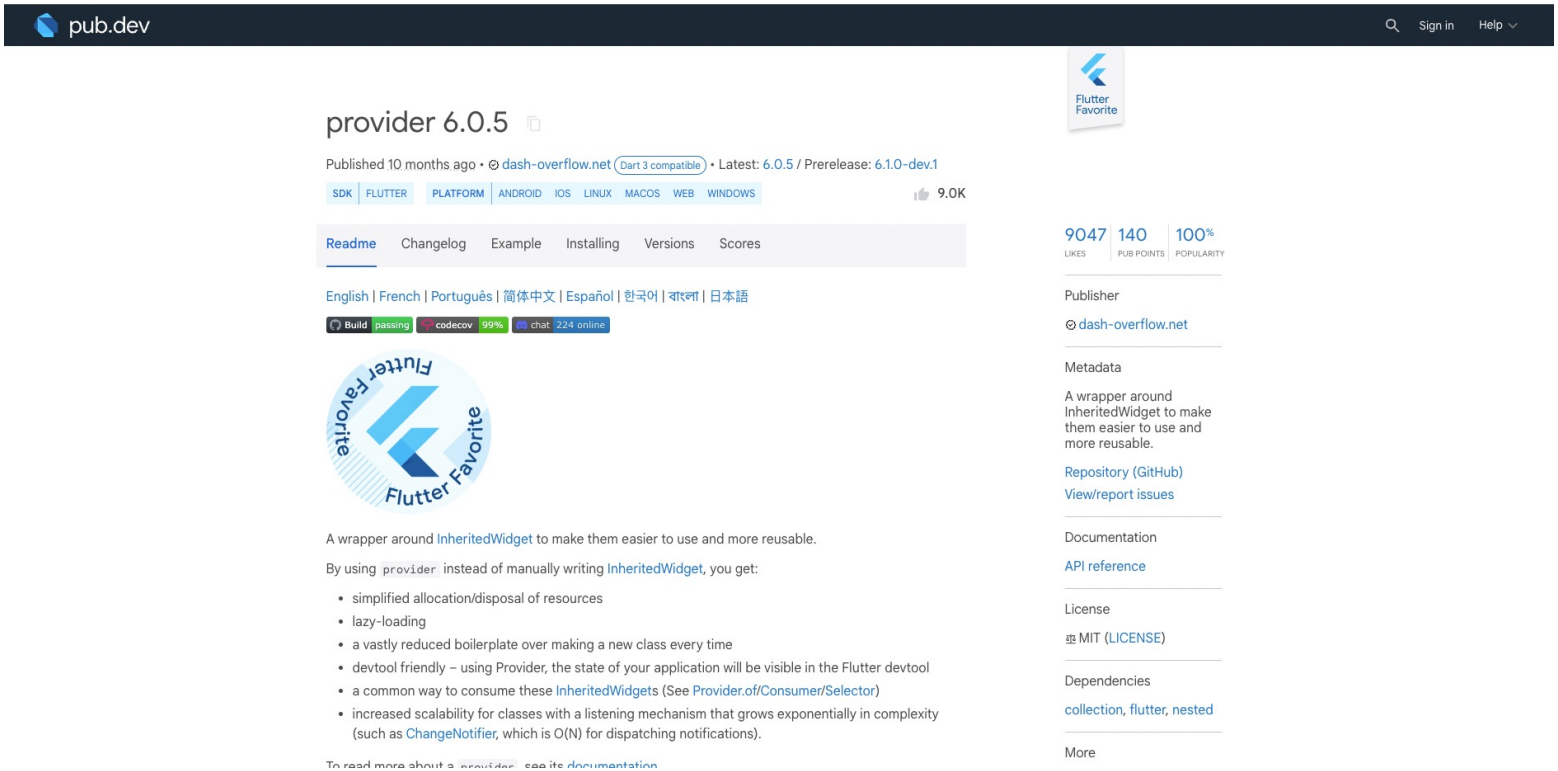
pub.dev


Sign in
Help
pub.dev
Search packages
The official package repository for Dart and Flutter apps.
Supported by Google
Flutter Favorites
Some of the packages that demonstrate the highest levels of quality, selected by the Flutter Ecosystem Committee
firebase_auth
Flutter plugin for Firebase Auth, enabling authentication using passwords, phone
firebase.google.com
msix
A command-line tool that create Msix installer from your flutter windows-build files.
kremer.software
auto_size_text
Flutter widget that automatically resizes text to fit perfectly within its bounds.
simc.dev
built_value_generator
Value types with builders, Dart classes as enums, and serialization. This library is the
google.dev

Adding a package

1. Search for a package

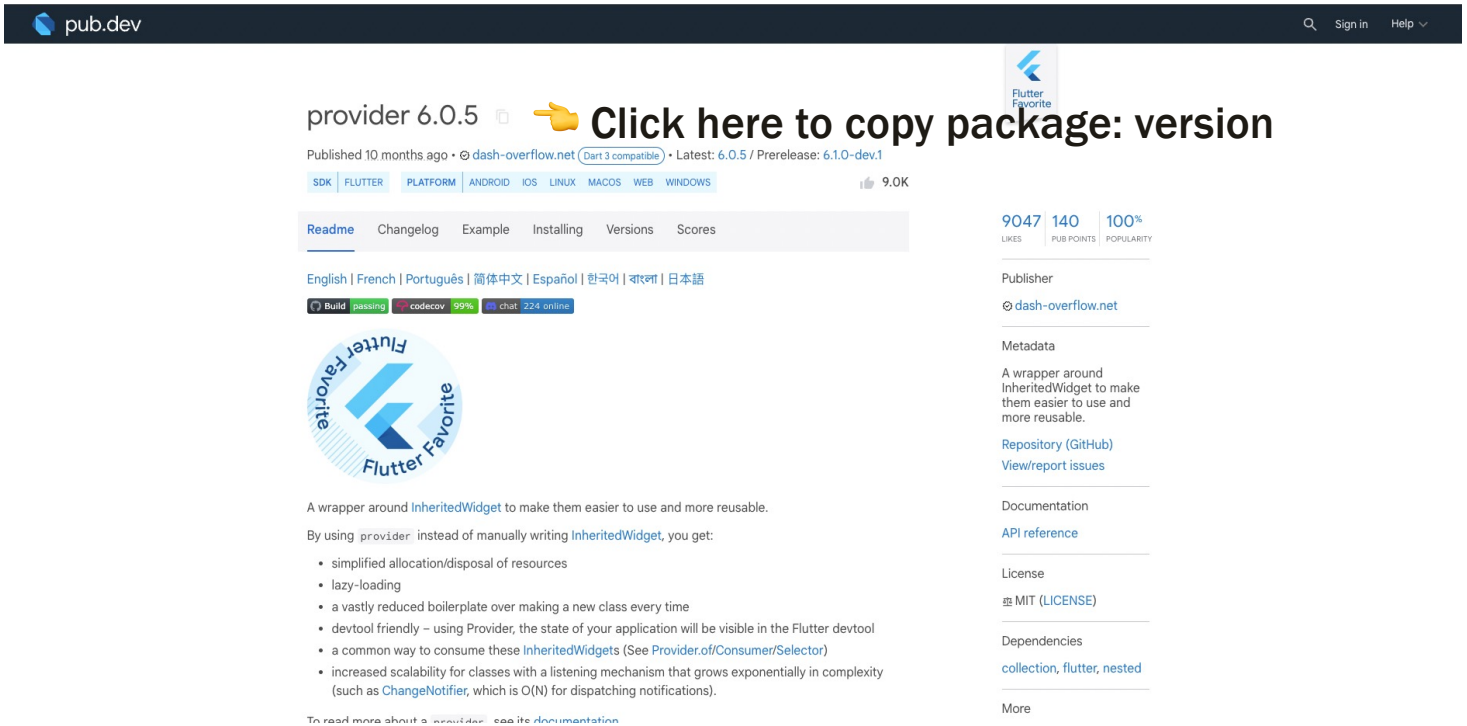
Searching for a package



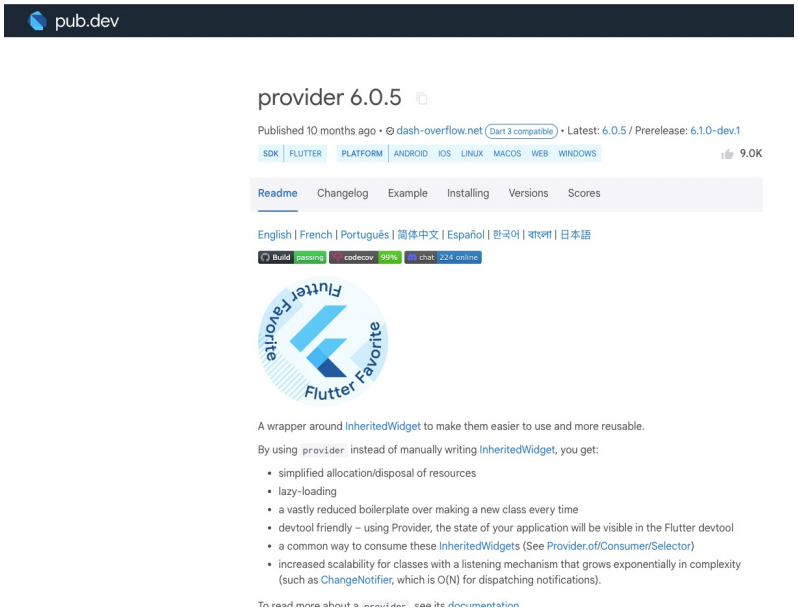
Adding a package

1. Search for a package
2. Add the package to the project

Adding package to project

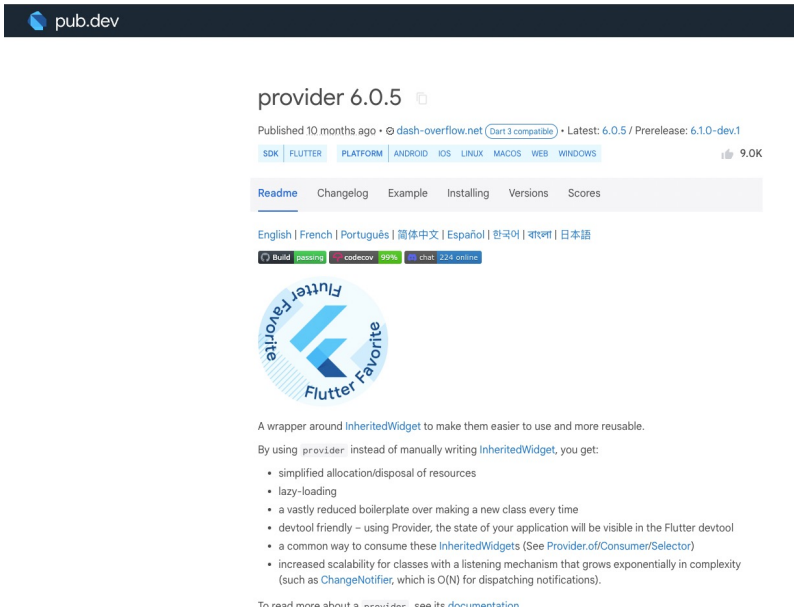


Adding package to project



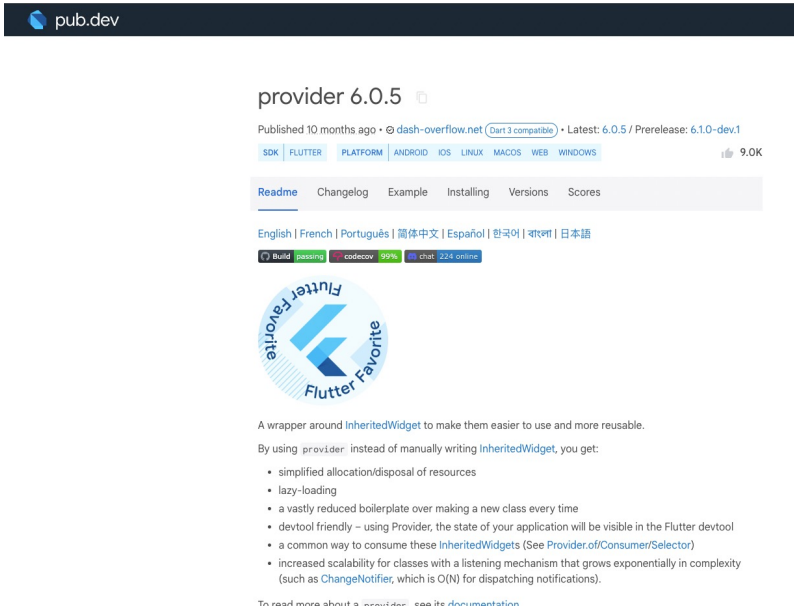

```
dependencies:
  flutter:
    sdk: flutter
  cupertino_icons: ^1.0.2
```

Adding package to project

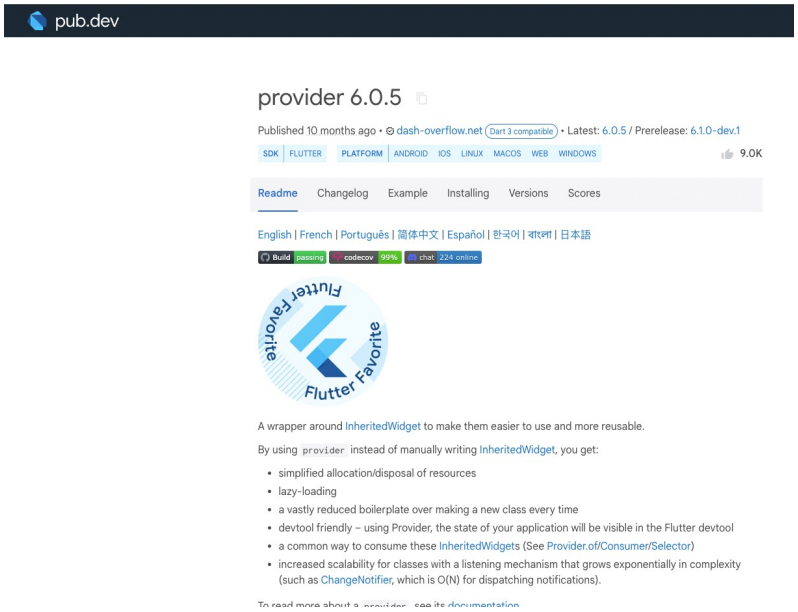



```
dependencies:
  flutter:
    sdk: flutter
  cupertino_icons: ^1.0.2
  provider: ^6.0.5
```

Adding package to project



Adding package to project



Adding a package

1. Search for a package
2. Add the package to the project
3. Resolve & fetch dependencies

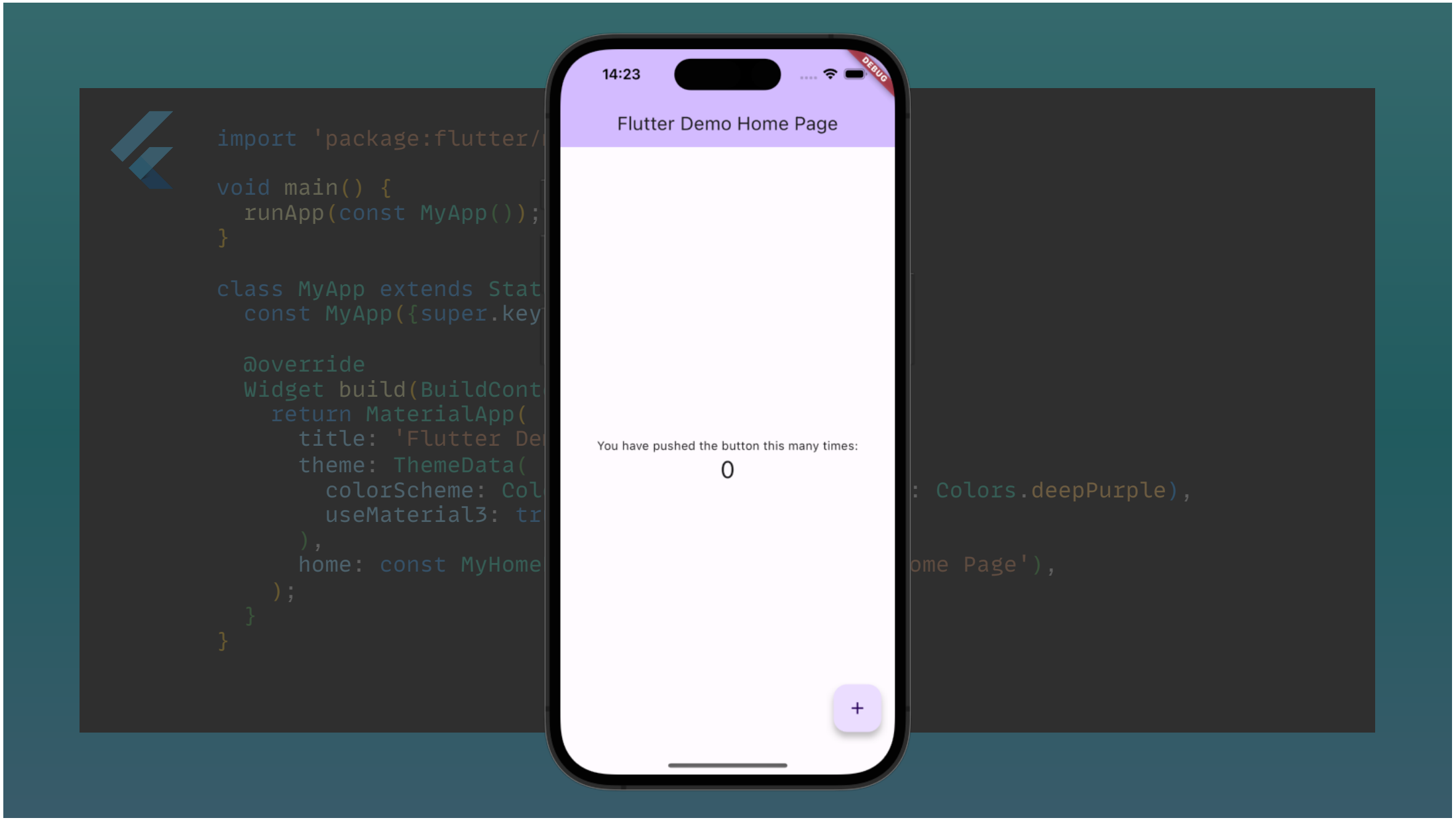
Resolving & fetching dependencies


ONE DOES NOT SIMPLY
UPGRADE THE DEPENDENCY VERSIONS
imgflip.com

Getting Started

Shopping List App

```dart
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}
```


import 'package:flutter/
void main() {
runApp(const MyApp());
}
class MyApp extends Stat
const MyApp({super.key
@override
Widget build(BuildCont
return MaterialApp(
title: 'Flutter De
theme: ThemeData(
colorScheme: Col
useMaterial3: tr
),
home: const MyHome
);
}
}
: Colors.deepPurple),
ome Page'),
14:23
DEBUG
Flutter Demo Home Page
You have pushed the button this many times:
0
+

```dart
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}
```

👈 Gives access to prebuilt widgets according to Material-UI Guidelines

👈 Entry point of the application

👈 Extending from StatelessWidget to define a stateless widget

👈 What is this?

MaterialApp

"A convenience widget that wraps a number of widgets that are commonly required for Material Design applications. It builds upon a WidgetsApp by adding material-design specific functionality [...]"

Types of apps

MaterialApp

- Provides styles to material widgets according to the Material design language for iOS, Android and web
- Most commonly used

CupertinoApp

- Provides styles to cupertino widgets based on Apple‘s Human Interface Guidelines
- Used to build standard iOS-styled apps

WidgetsApp

- Basis for the other two apps
- Rarely to never used by actual Flutter developers

```dart
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}
```

👈 Defines the theme of the application

👆 Defines the page that is initially rendered

```dart
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(
          seedColor: const Color(
            0xff38bdf8,
          ),
        ),
        useMaterial3: true,
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}
```

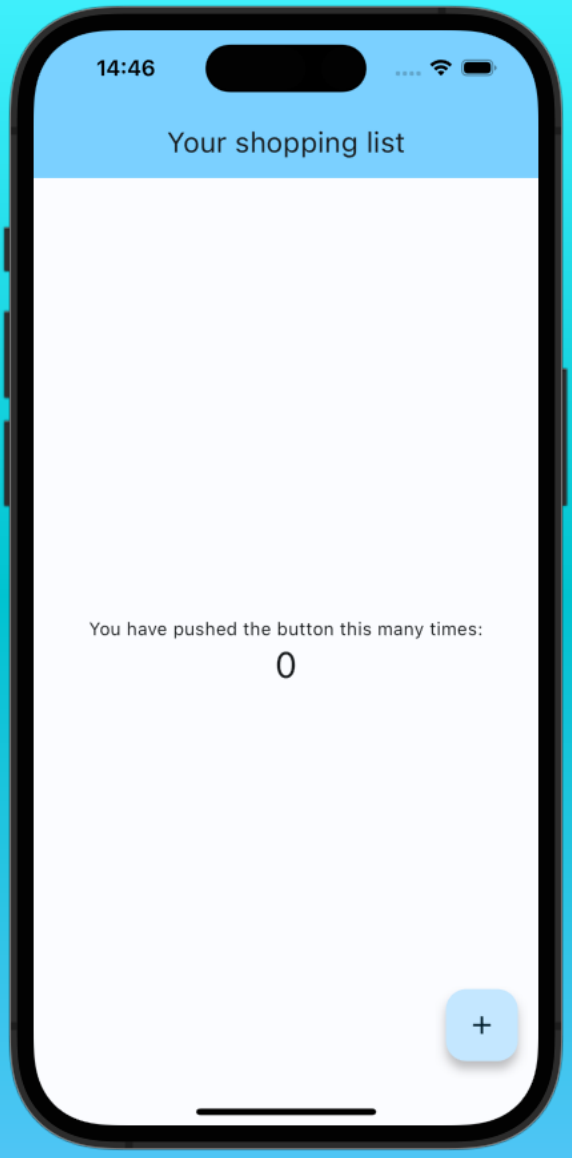
👈 Custom hex color 🟦 prefixed with 0xff

How do I run the app?

Using the flutter cli

Running the app

- Looks for any connected device (simulators, physical devices, ...)
- When started, type “r“ to hot reload and “R“ to hot restart

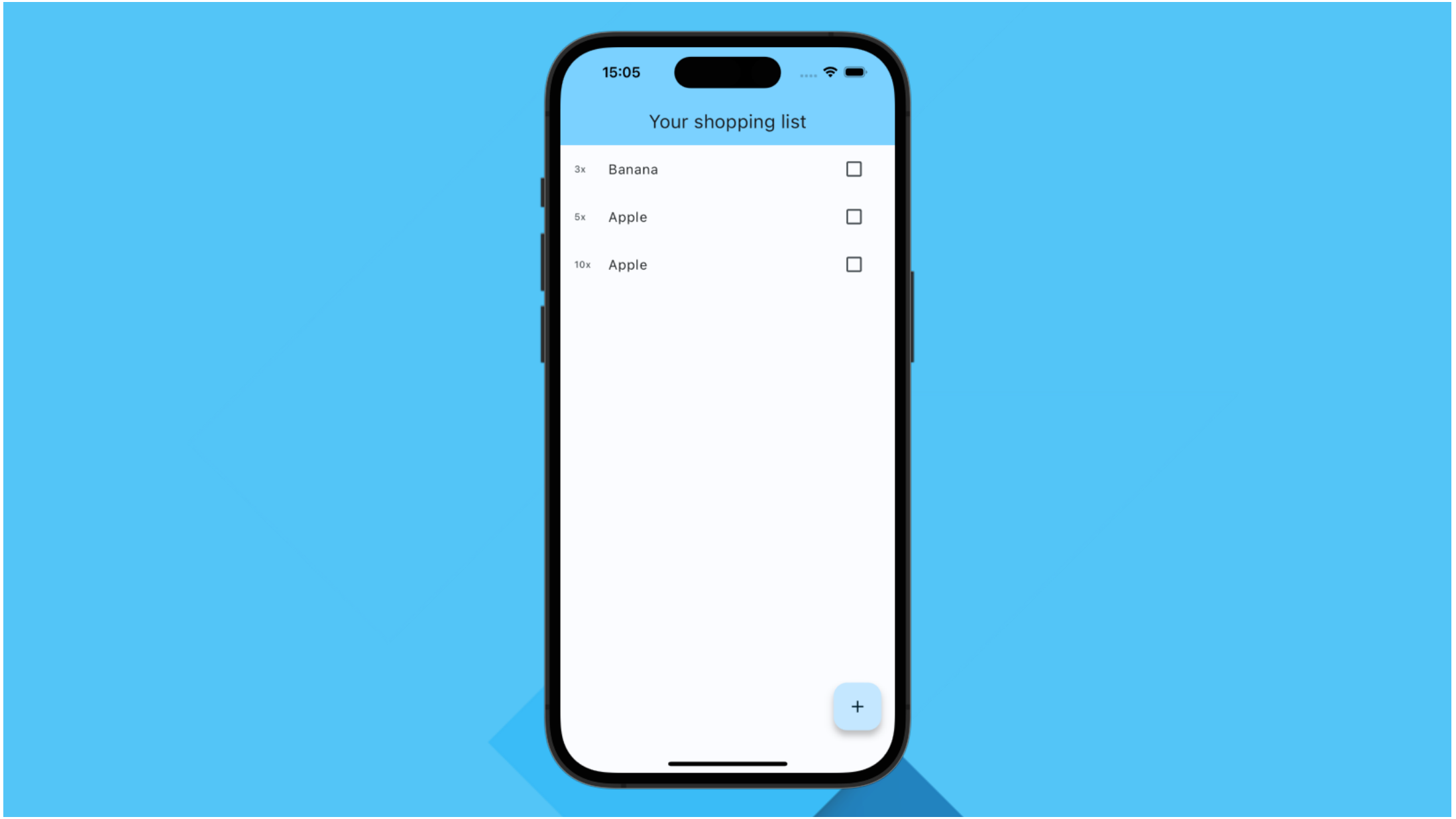


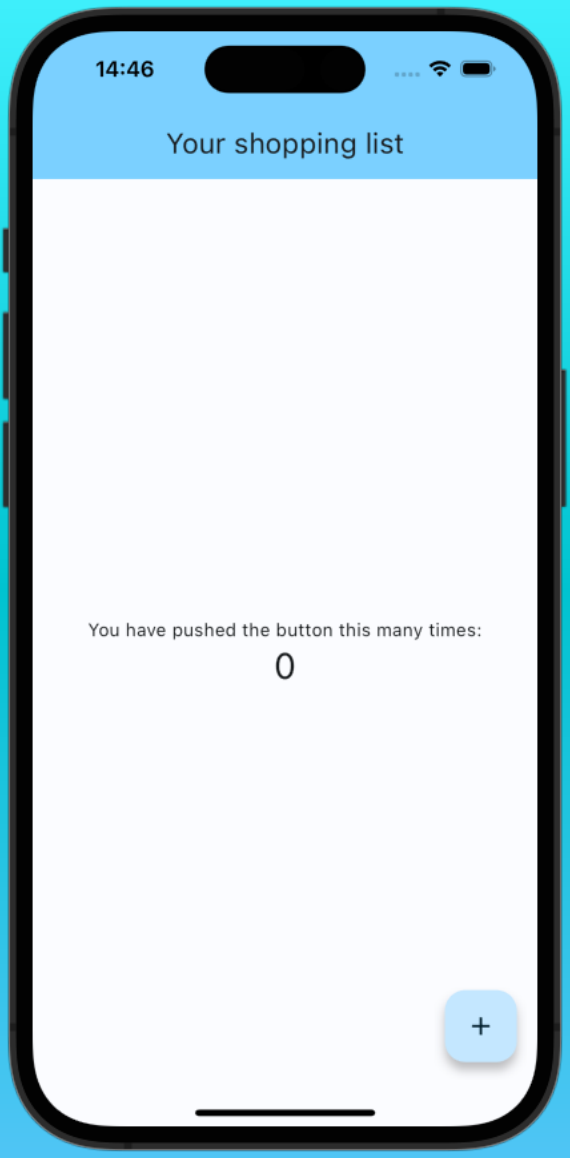

```dart
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Shopping list',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(
          seedColor: const Color(
            0xff38bdf8,
          ),
        ),
        useMaterial3: true,
      ),
      home: const MyHomePage(title: 'Your shopping list'),
    );
  }
}
```

👆 Hide DEBUG banner

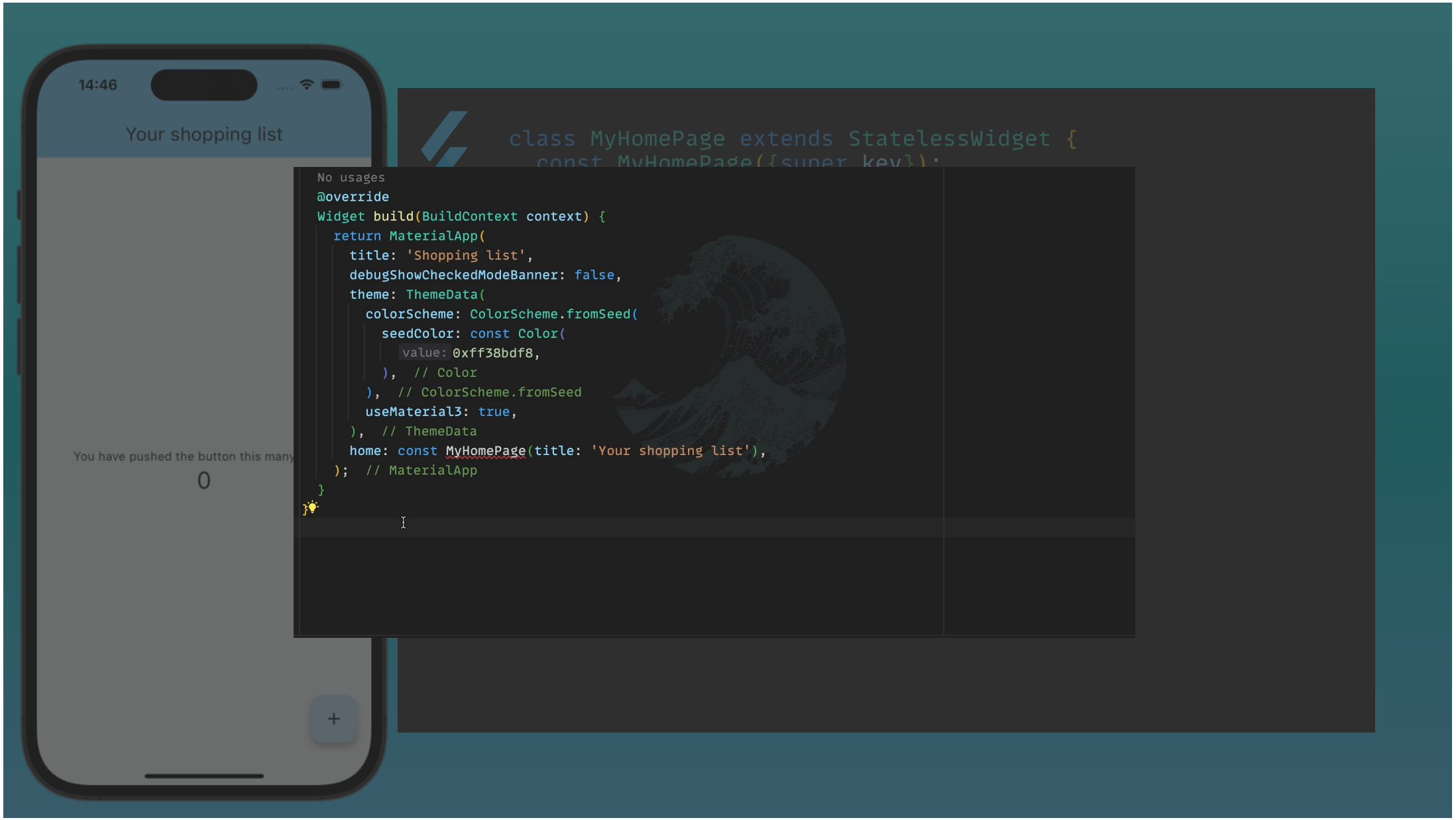
Shopping list app

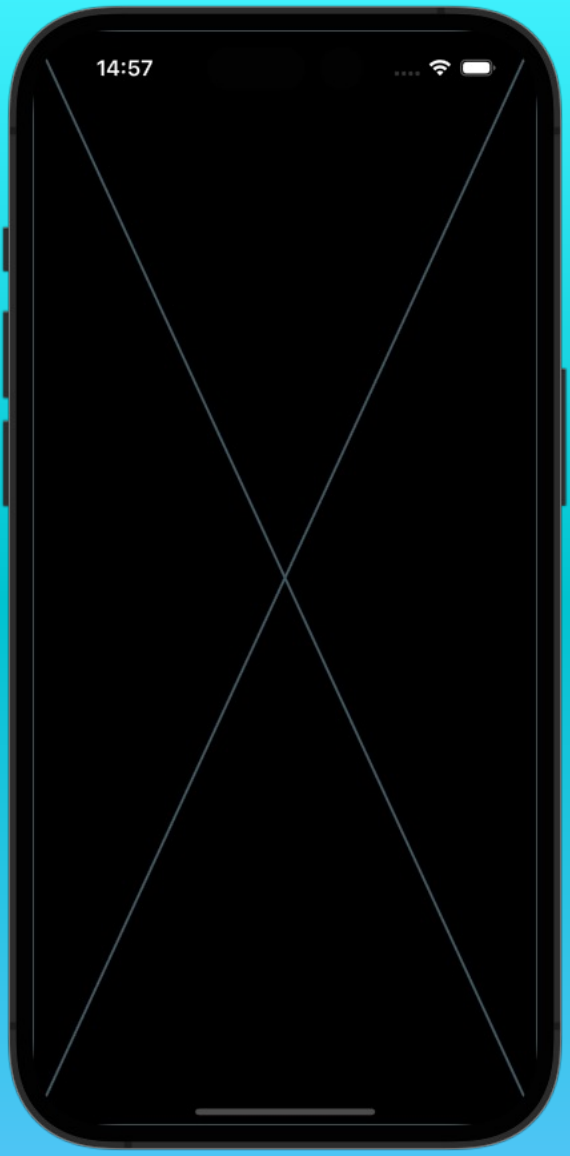
Creating the first page


15:05
Your shopping list
3x Banana
5x Apple
10x Apple
+




```dart
class MyHomePage extends StatelessWidget {
  const MyHomePage({super.key});

  @override
  Widget build(BuildContext context) {
    return const Placeholder();
  }
}
```


14:46
Your shopping list
You have pushed the button this many
0
class MyHomePage extends StatelessWidget {
No usages
@override
Widget build(BuildContext context) {
return MaterialApp(
title: 'Shopping list',
debugShowCheckedModeBanner: false,
theme: ThemeData(
colorScheme: ColorScheme.fromSeed(
seedColor: const Color(
value: 0xff38bdf8,
), // Color
), // ColorScheme.fromSeed
useMaterial3: true,
), // ThemeData
home: const MyHomePage(title: 'Your shopping list'),
); // MaterialApp
}
}



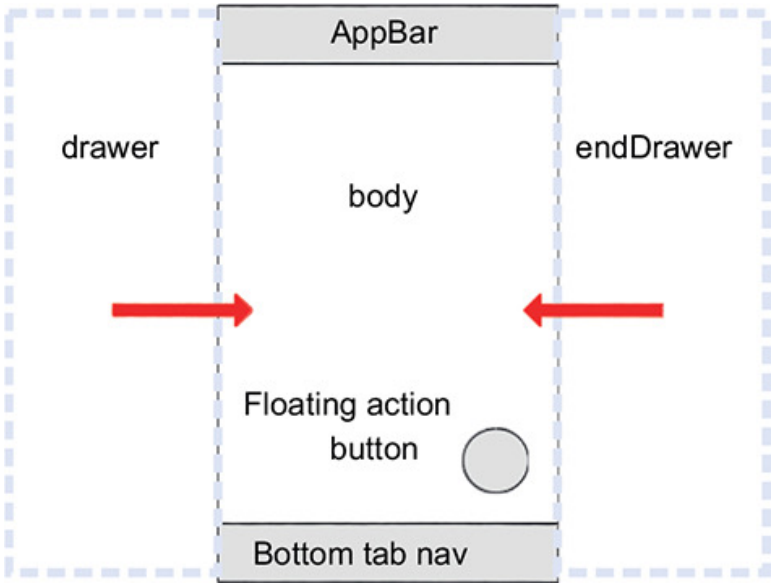

```dart
class MyHomePage extends StatelessWidget {
  const MyHomePage({super.key, required this.title});

  final String title;

  @override
  Widget build(BuildContext context) {
    return const Placeholder();
  }
}
```

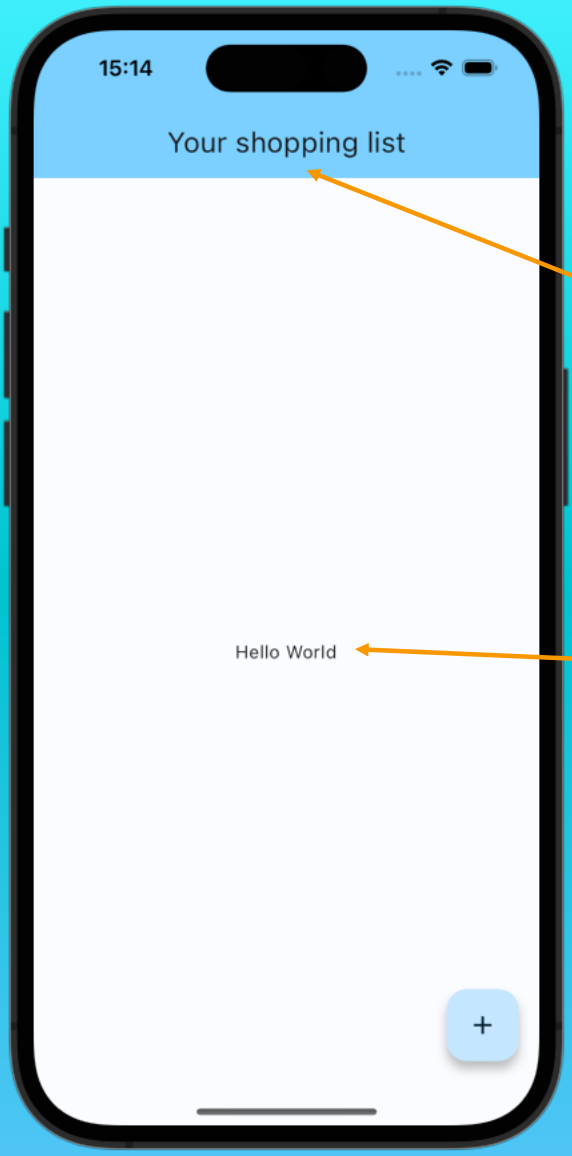
Scaffold

"Implements the basic Material Design visual layout structure."

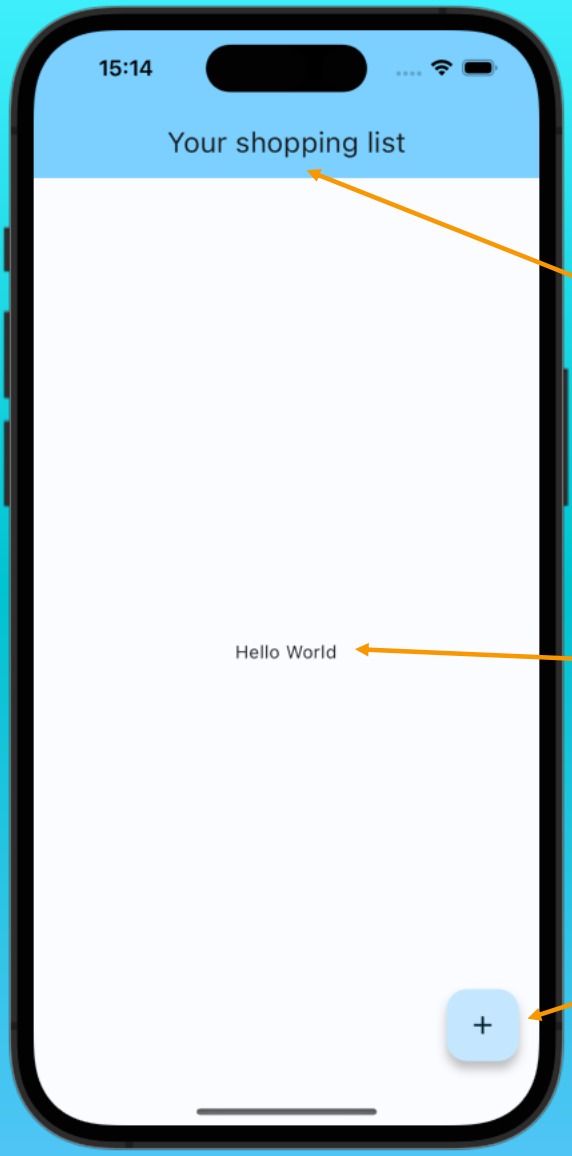
Scaffold

- appBar: “An app bar to display at the top of the scaffold.“
- body: “The primary content of the scaffold.“
- floatingActionButton: “A button displayed floating above body, in the bottom right corner.“
- backgroundColor: “The color of the Material widget that underlies the entire scaffold.“
- see all properties here






```dart
class MyHomePage extends StatelessWidget {
  const MyHomePage({super.key, required this.title});

  final String title;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
        title: Text(title),
      ),
      body: const Center(
        child: Text('Hello World'),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {},
        child: const Icon(Icons.add),
      ),
    );
  }
}
```




```dart
class MyHomePage extends StatelessWidget {
  const MyHomePage({super.key, required this.title});

  final String title;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
        title: Text(title),
      ),
      body: const Center(
        child: Text('Hello World'),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {},
        child: const Icon(Icons.add),
      ),
    );
  }
}
```

Center

“A widget that centers its child within itself.“

Center

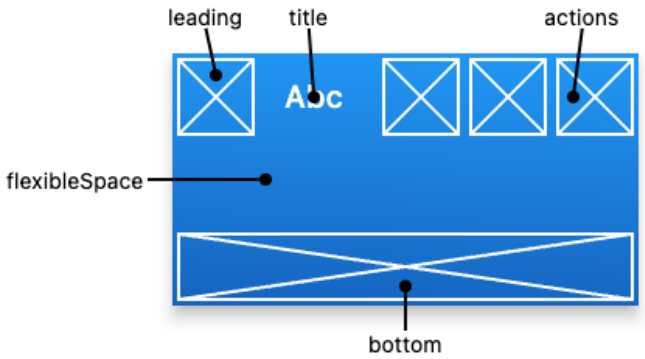
“A widget that centers its child within itself.“

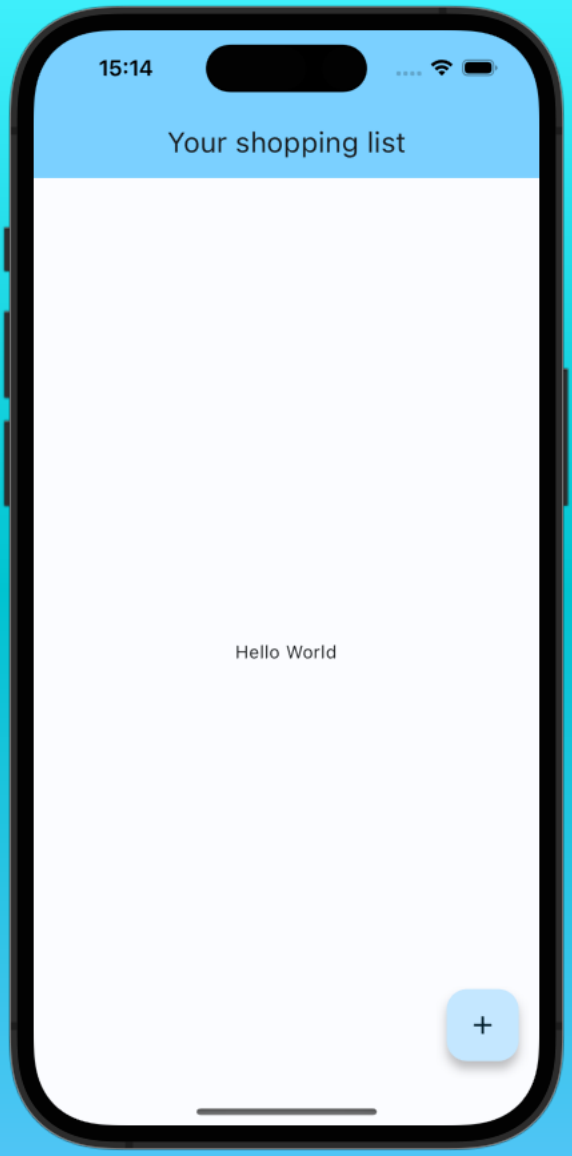
AppBar

"A Material Design app bar."

AppBar

- title: “The primary widget displayed in the app bar.“
- actions: “A list of Widgets to display in a row after the title widget.“
- leading: “A widget to display before the toolbar‘s title.“
- bottom: “This widget appears across the bottom of the app bar.“
- see all properties here

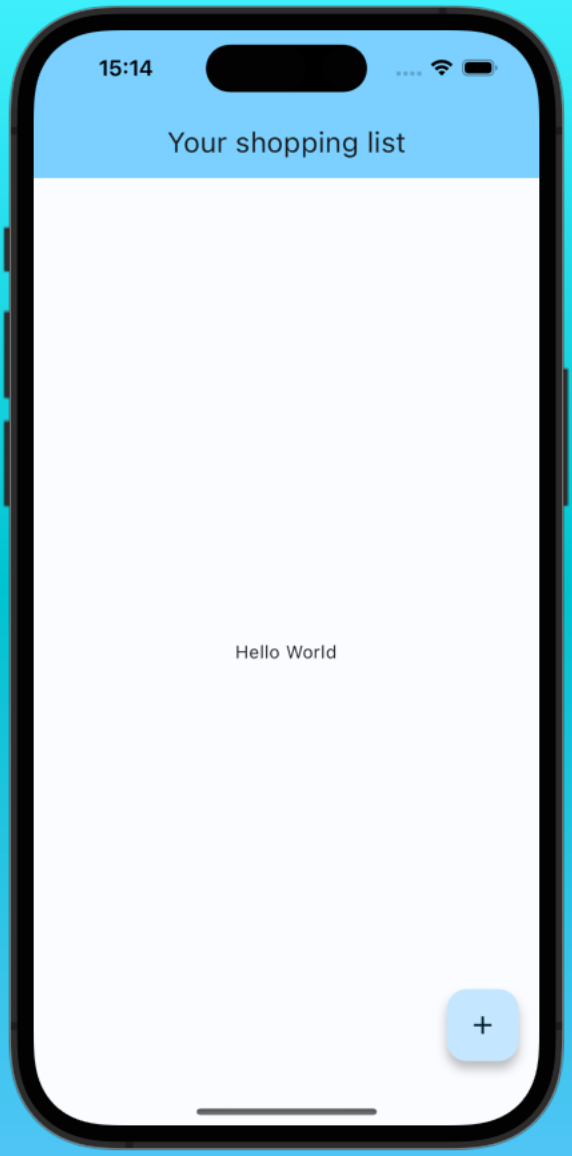




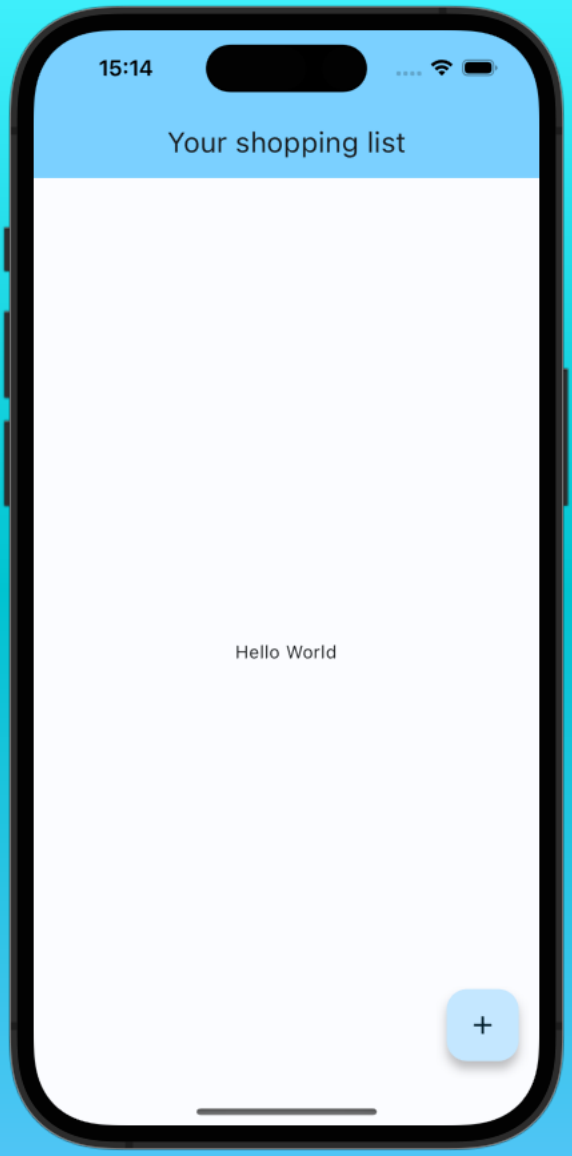

```dart
class MyHomePage extends StatelessWidget {
  const MyHomePage({super.key, required this.title});

  final String title;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
        title: Text(title),
      ),
      body: const Center(
        child: Text('Hello World'),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {},
        child: const Icon(Icons.add),
      ),
    );
  }
}
```

Shopping list app

Defining the models



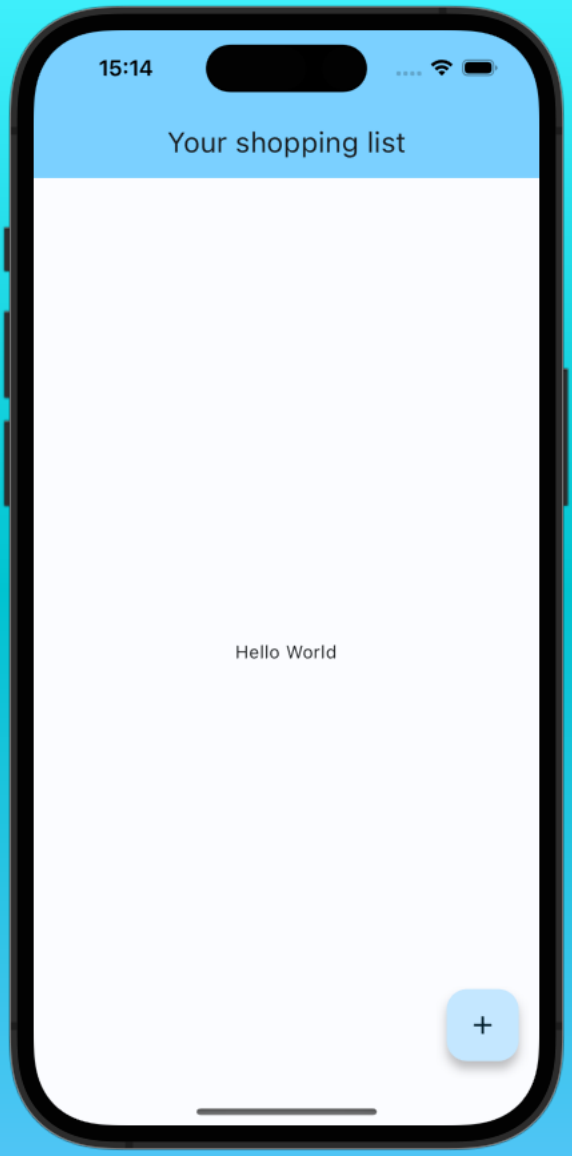

```dart
class ShoppingList {
  final List<ShoppingListItem> items;

  ShoppingList() : items = [];
}

class ShoppingListItem {
  final int id;
  final String name;
  final int quantity;
  bool isBought;

  ShoppingListItem({
    required this.id,
    required this.name,
    required this.quantity,
    required this.isBought,
  });
}
```




```dart
class ShoppingList {
  final List<ShoppingListItem> items;

  static int _nextId = 0;

  ShoppingList() : items = [];

  void addItem({
    required String name,
    required int quantity,
  }) {
    items.add(
      ShoppingListItem(
        id: _nextId++,
        name: name,
        quantity: quantity,
        isBought: false,
      ),
    );
  }
}
```

List of ShoppingListItems

Shopping List App



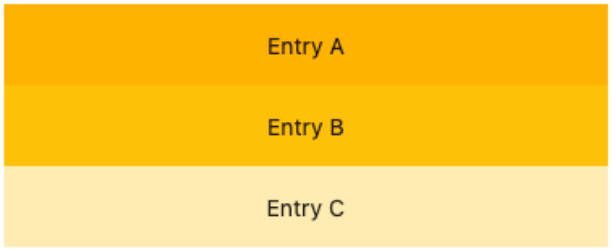

```dart
class ShoppingListView extends StatefulWidget {
  const ShoppingListView({super.key});

  @override
  State<ShoppingListView> createState() =>
_ShoppingListViewState();
}

class _ShoppingListViewState extends State<ShoppingListView>
{
  @override
  Widget build(BuildContext context) {
    return const Placeholder();
  }
}
```

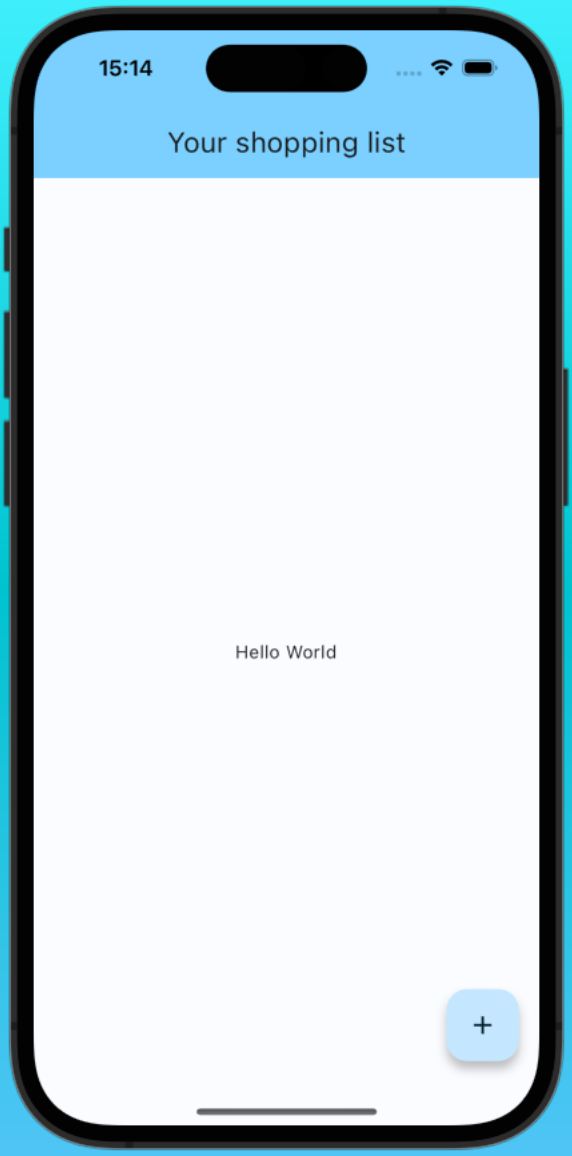
ListView

"A scrollable list of widgets arranged linearly."

ListView

- children: The widgets layed out along the main axis.
- scrollDirection: “The axis along which the scroll view‘s offset increases.“
- shrinkWrap: “Whether the extent of the scroll view in the scrollDirection should be determined by the contents being viewed.“
- controller: “An object that can be used to control the position to which this scroll view is scrolled.“
- see all properties here





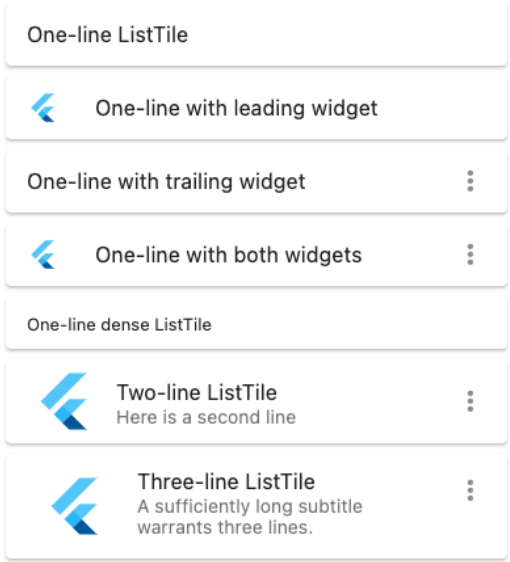

```dart
class _ShoppingListViewState extends State<ShoppingListView>
{
  ShoppingList _shoppingList = ShoppingList();

  @override
  Widget build(BuildContext context) {
    return ListView(
      children: [
        for (final item in _shoppingList.items)
          ListTile(
            leading: Text(item.id.toString()),
            title: Text(item.name),
          )
      ],
    );
  }
}
```

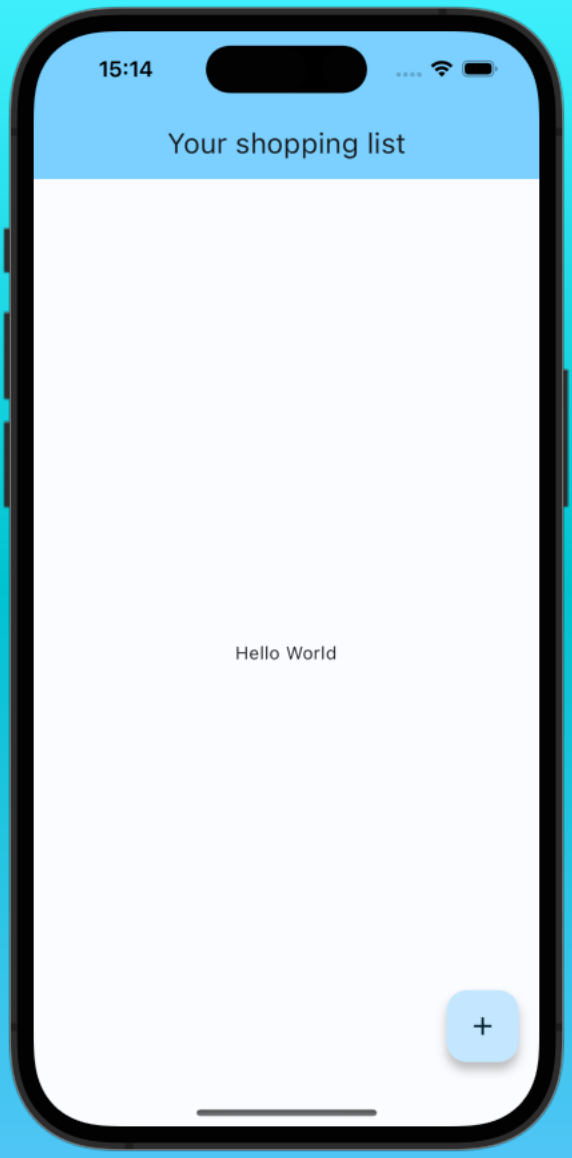
ListTile

“A single fixed-height row that typically contains some text as well as a leading or trailing icon.“

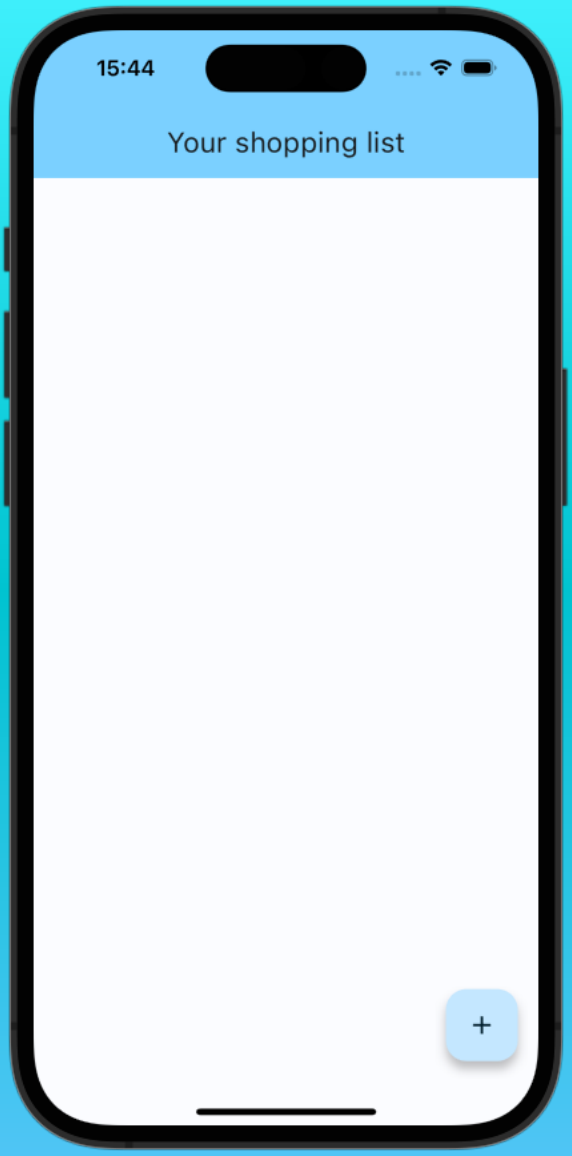
ListTile

- title: “The primary content of the list tile.“
- leading: “A widget to display before the title.“
- trailing: “A widget to display after the title.“
- subtitle: “Additional content displayed below the title.“
- onTap: “Called when the user taps this list tile.“
- see all properties here

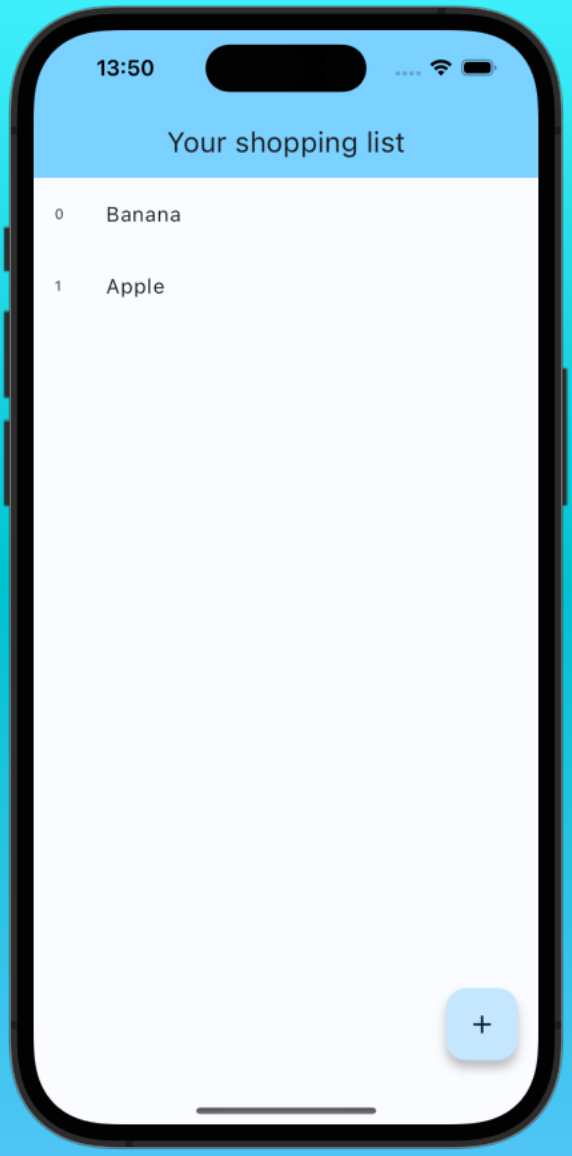




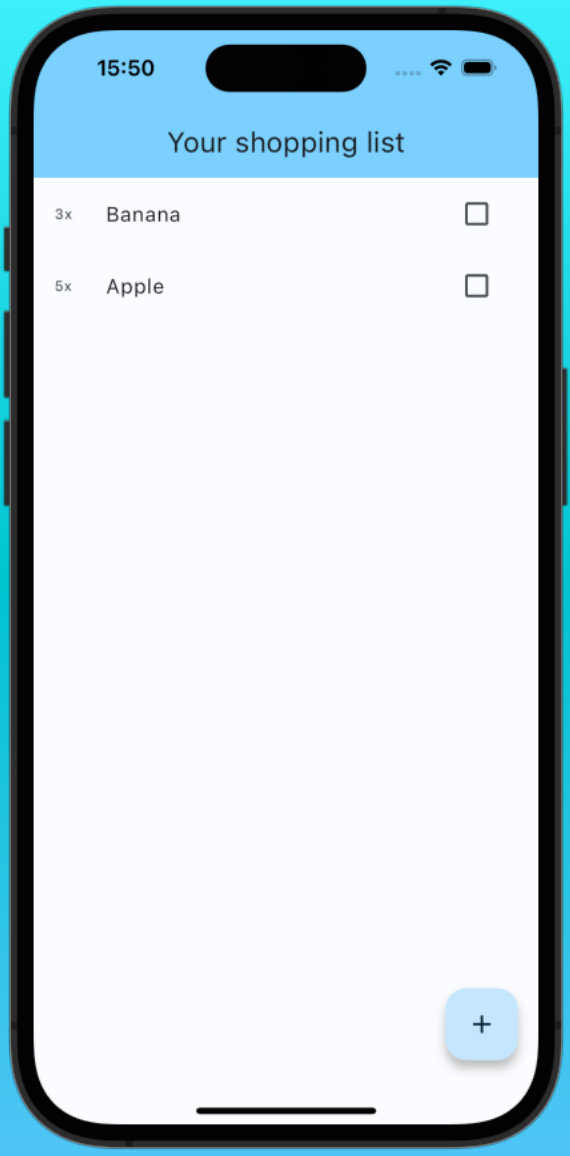

```dart
class _ShoppingListViewState extends State<ShoppingListView>
{
  ShoppingList _shoppingList = ShoppingList();

  @override
  Widget build(BuildContext context) {
    return ListView(
      children: [
        for (final item in _shoppingList.items)
          ListTile(
            leading: Text(item.id.toString()),
            title: Text(item.name),
          )
      ],
    );
  }
}
```




```dart
class MyHomePage extends StatelessWidget {
  const MyHomePage({super.key, required this.title});

  final String title;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor:
          Theme.of(context).colorScheme.inversePrimary,
        title: Text(title),
      ),
      body: const ShoppingListView(),
      floatingActionButton: FloatingActionButton(
        onPressed: () {},
        child: const Icon(Icons.add),
      ),
    );
  }
}
```




```dart
class _ShoppingListViewState extends State<ShoppingListView>
{
  ShoppingList _shoppingList = ShoppingList()
    ..addItem(
      name: 'Banana',
      quantity: 3,
    )
   ..addItem(
      name: 'Apple‘,
      quantity: 5);

  @override
  Widget build(BuildContext context) {
    return ListView(
      children: [
        for (final item in _shoppingList.items)
          ListTile(
            leading: Text(item.id.toString()),
            title: Text(item.name),
          )
      ],
    );
  }
}
```



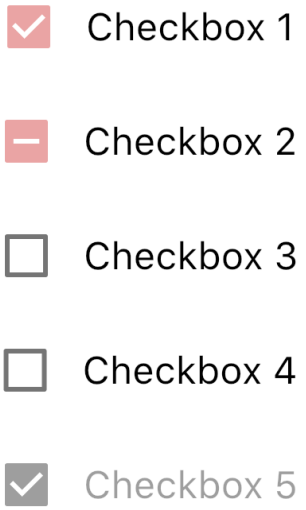
```dart
@override
Widget build(BuildContext context) {
  return ListView(
    children: [
      for (final item in _shoppingList.items)
        ListTile(
          leading: Text('${item.quantity}x'),
          title: Text(item.name),
          trailing: Checkbox(
            value: false,
            onChanged: (value) {},
          ),
        )
    ],
  );
}
```

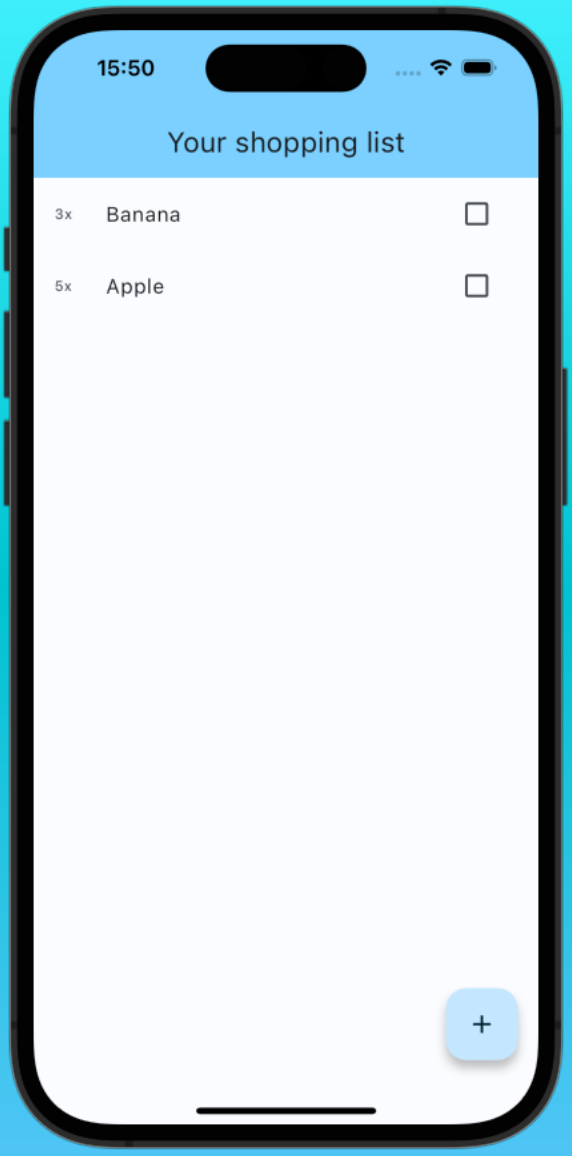
CheckBox

“A Material Design checkbox.“

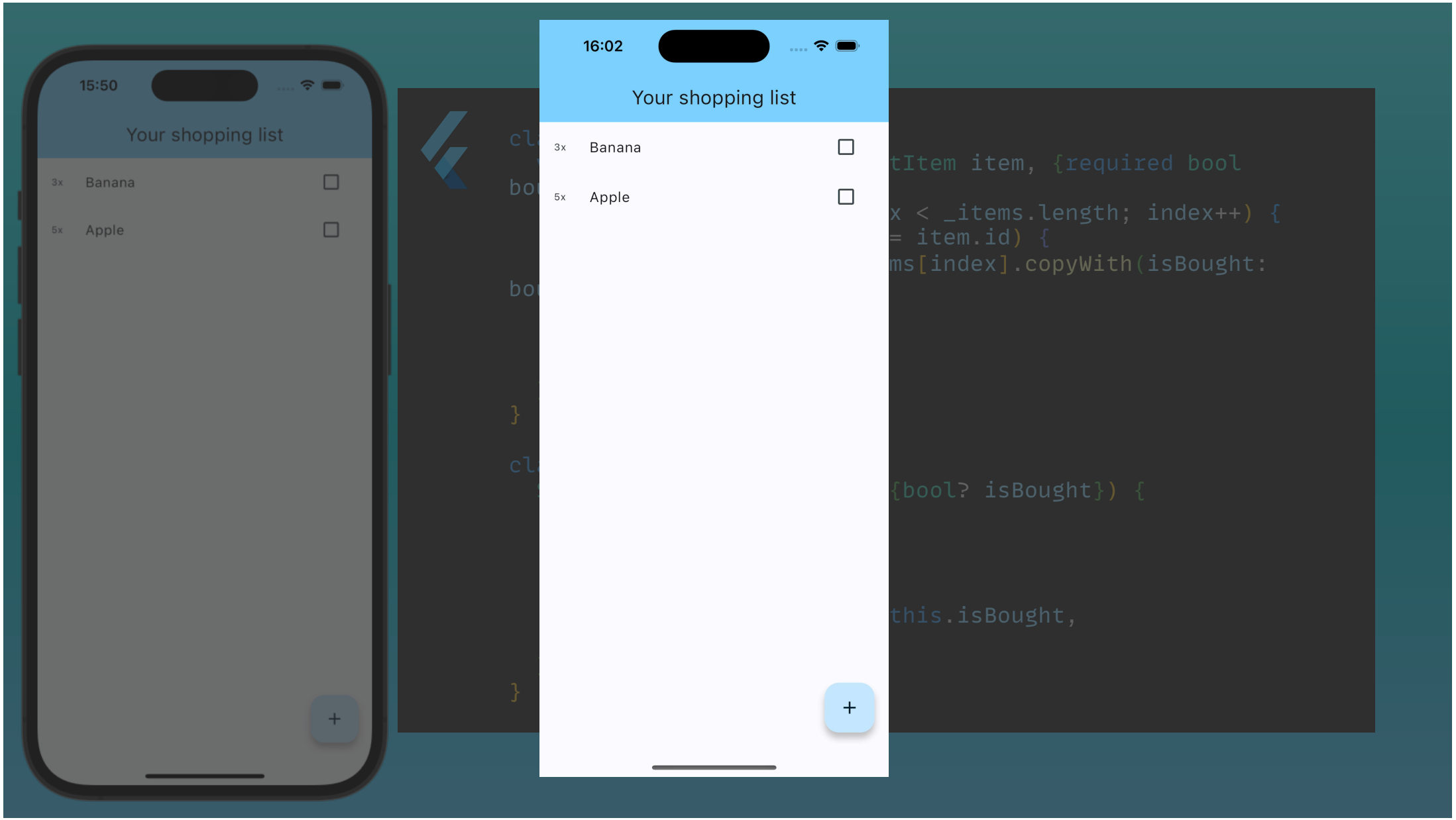
CheckBox

- value: “Whether this checkbox is checked.“
- onChanged: “Called when the value of the checkbox should change.“
- see all properties here



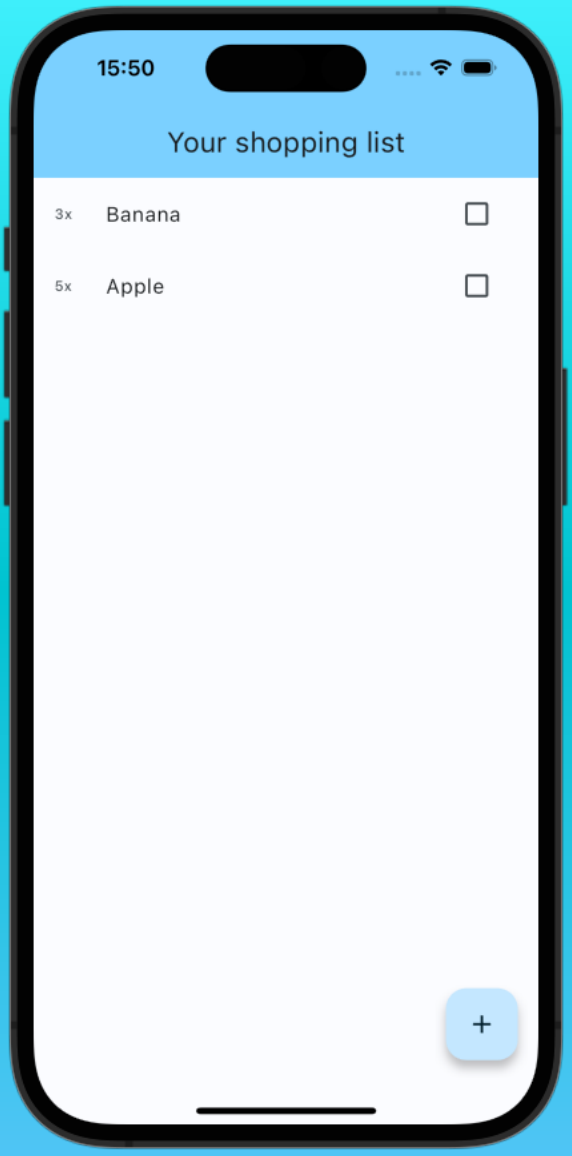



```dart
@override
Widget build(BuildContext context) {
  return ListView(
    children: [
      for (final item in _shoppingList.items)
        ListTile(
          leading: Text('${item.quantity}x'),
          title: Text(item.name),
          trailing: Checkbox(
            value: item.isBought,
            onChanged: (value) {
                item.isBought = value!;
            },
          ),
        )
    ],
  );
}
```


15:50
Your shopping list
3x Banana
5x Apple
16:02
Your shopping list
3x Banana
5x Apple

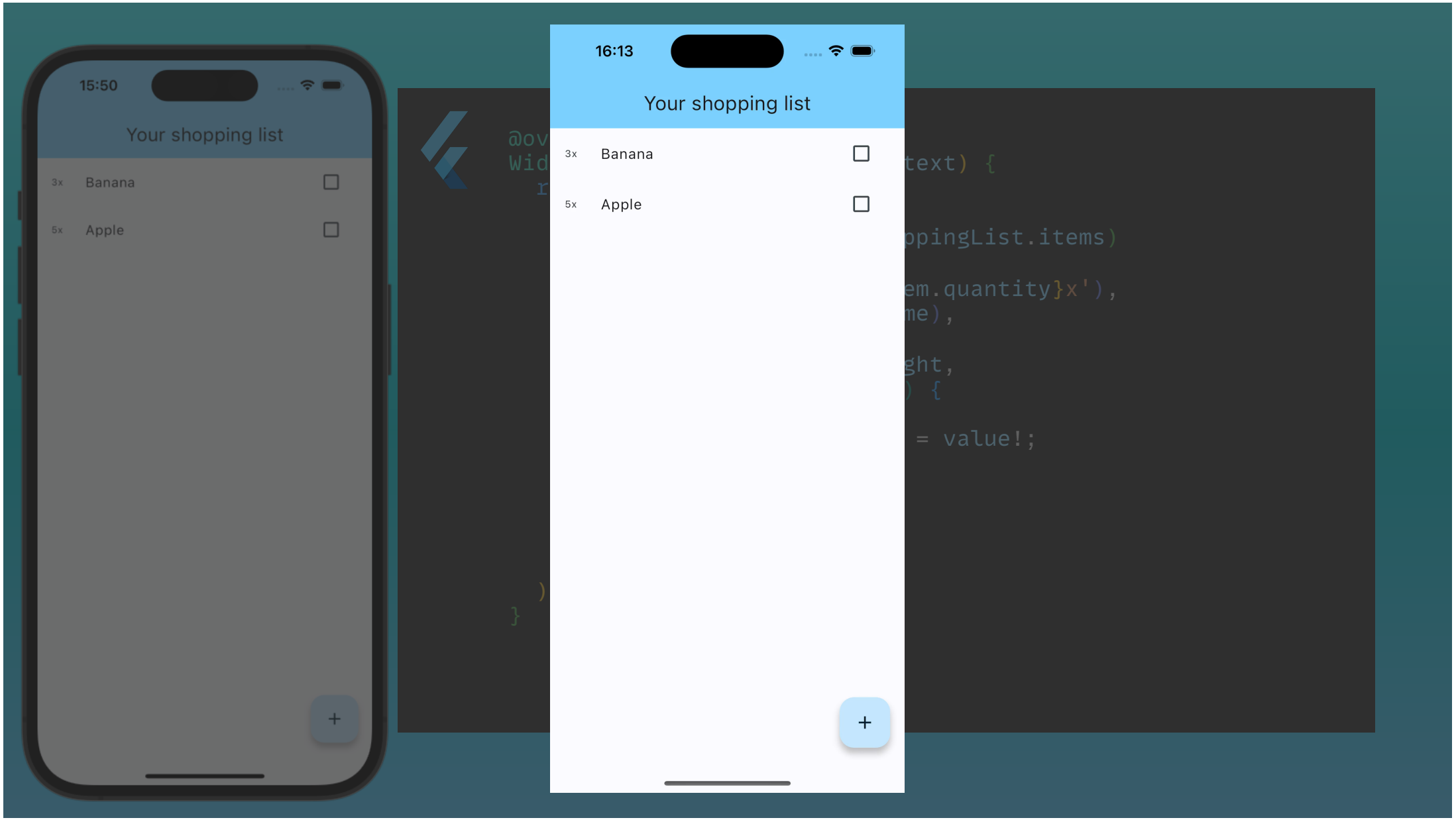
Simple State Management

Shopping List App



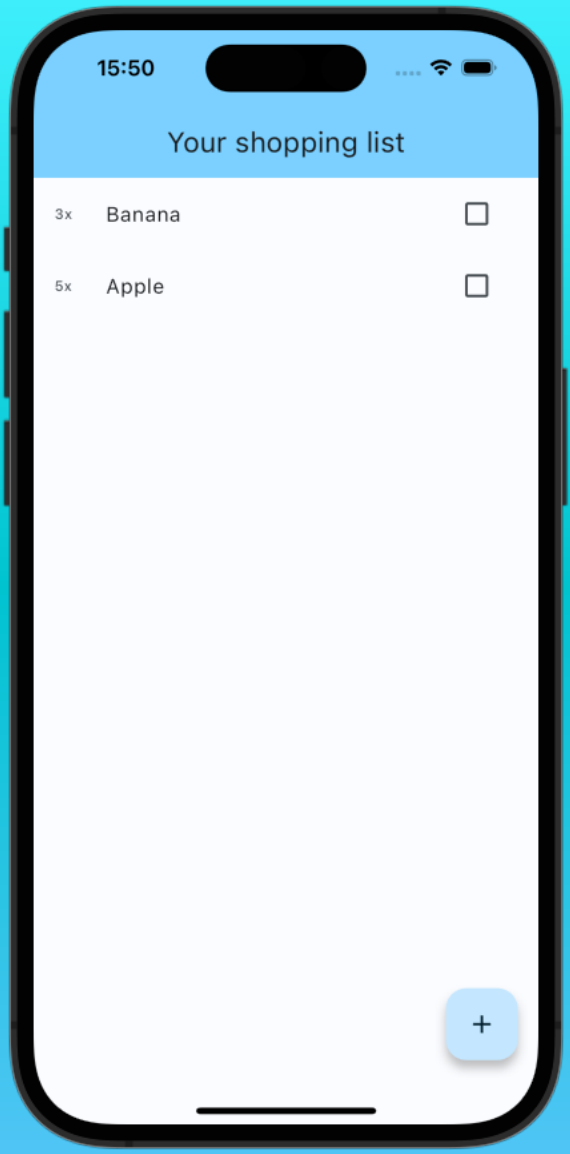

```dart
@override
Widget build(BuildContext context) {
  return ListView(
    children: [
      for (final item in _shoppingList.items)
        ListTile(
          leading: Text('${item.quantity}x'),
          title: Text(item.name),
          trailing: Checkbox(
            value: item.isBought,
            onChanged: (value) {
              setState(() {
                item.isBought = value!;
              });
            },
          ),
        )
    ],
  );
}
```

👆setState rerenders the whole widget sub tree


15:50
Your shopping list
3x Banana
5x Apple
16:13
Your shopping list
3x Banana
5x Apple

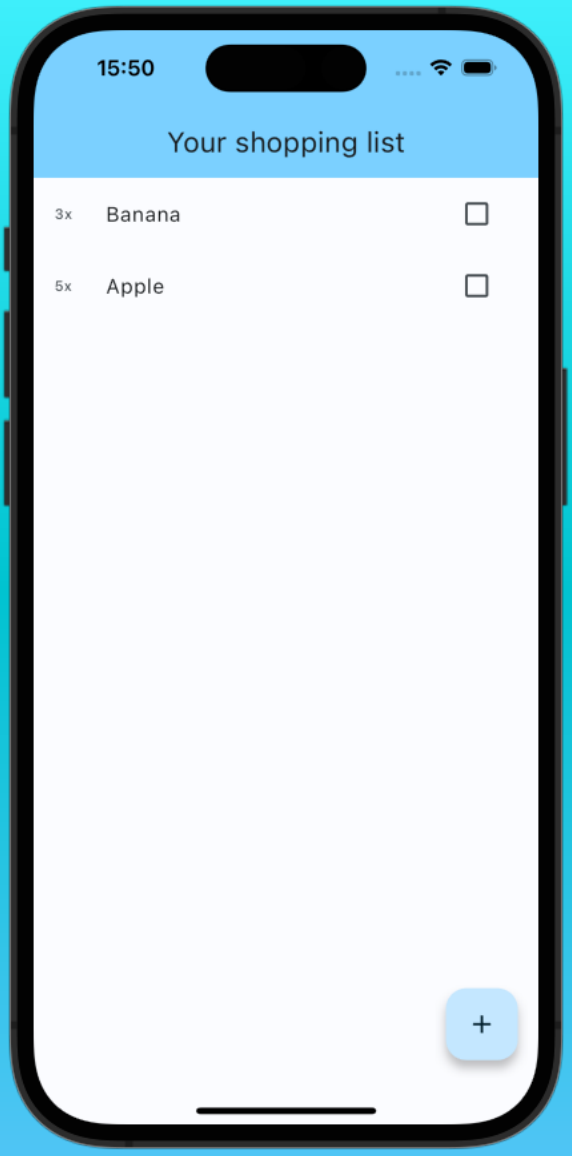
Toggling the Checkbox with ListTile click

ShoppingListApp

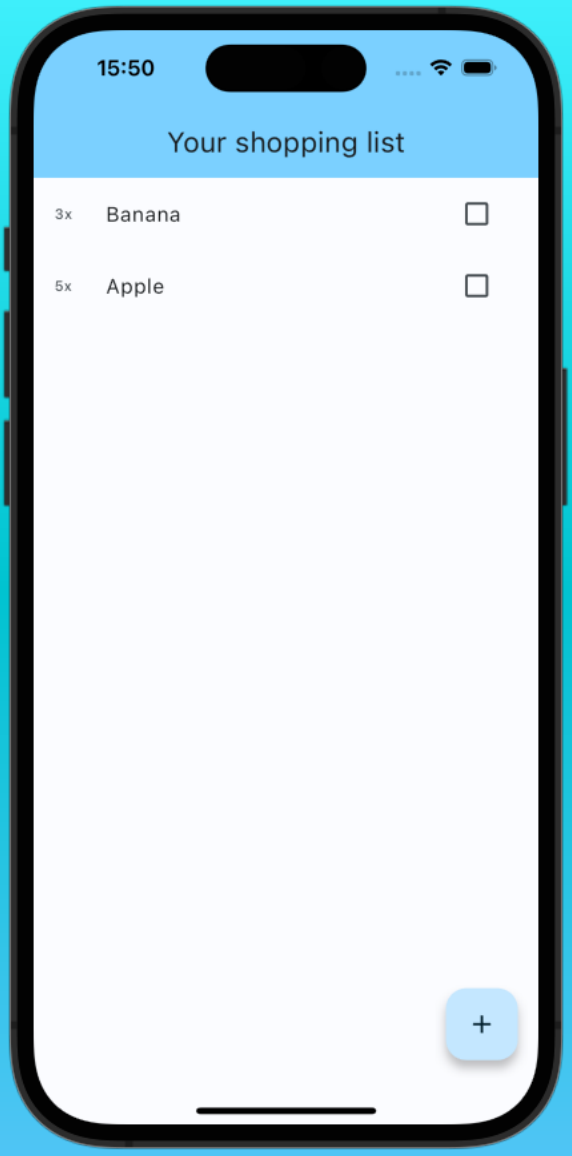



```dart
ListTile(
  onTap: () {
    setState(() {
      item.isBought = !item.isBought;
    });
  },
  leading: Text('${item.quantity}x'),
  title: Text(item.name),
  trailing: Checkbox(
    value: item.isBought,
    onChanged: (value) {
      setState(() {
        item.isBought = value!;
      });
    },
  ),
)
```

Duplicated code



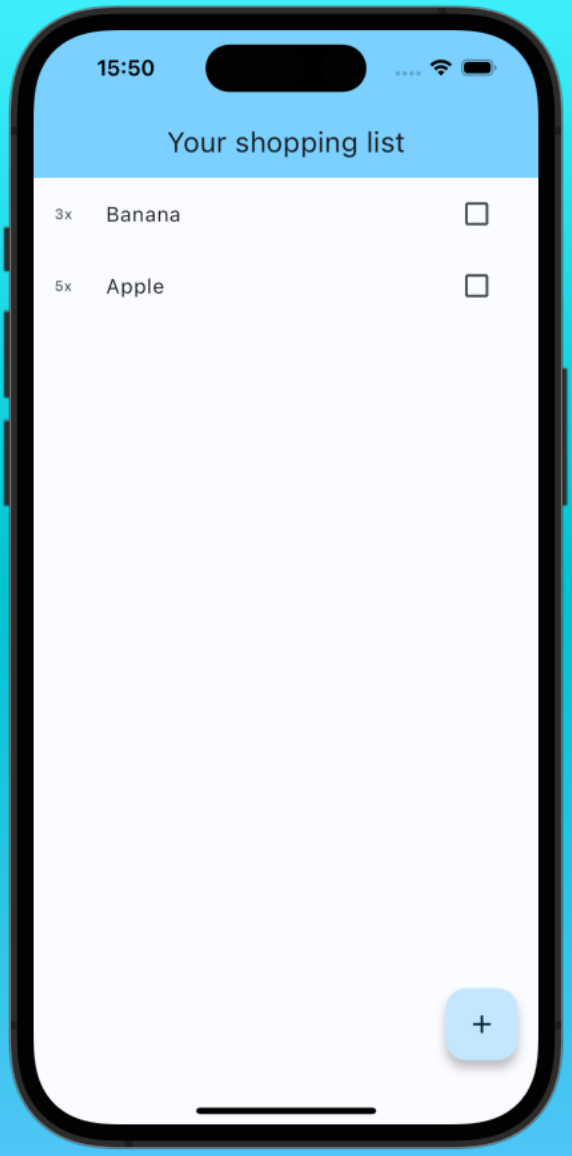

```dart
void _toggleBought(ShoppingListItem item) {
  setState(() {
    item.isBought = !item.isBought;
  });
}
```




```dart
ListTile(
  onTap: () {
    _toggleBought(item);
  },
  leading: Text('${item.quantity}x'),
  title: Text(item.name),
  trailing: Checkbox(
    value: item.isBought,
    onChanged: (_) {
      _toggleBought(item);
    },
  ),
)
```

Current Issues

Shopping List App



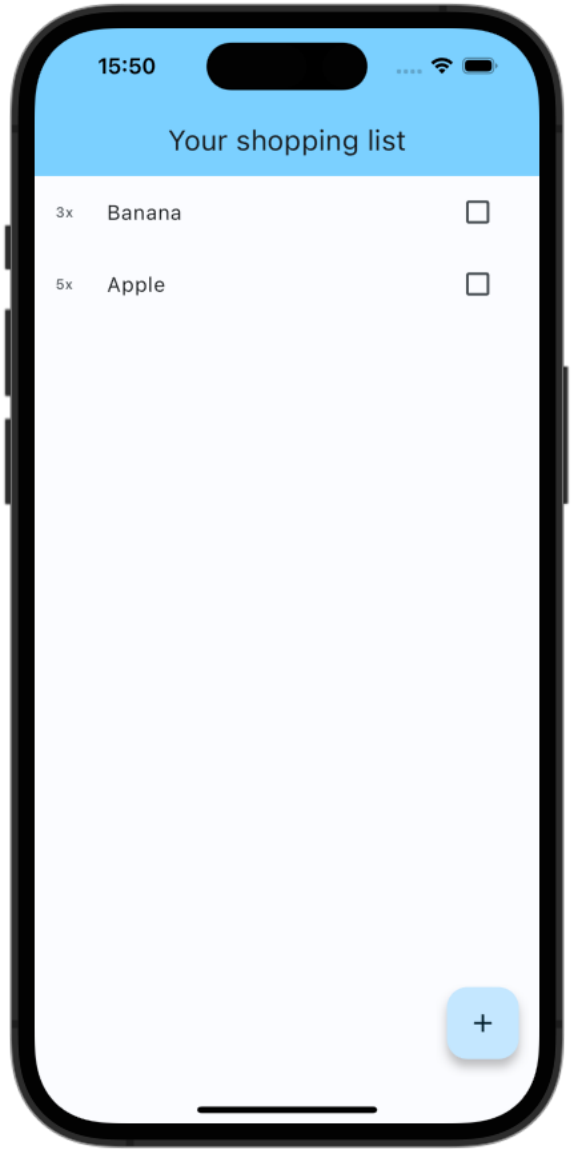

```dart
@override
Widget build(BuildContext context) {
  return ListView(
    children: [
      for (final item in _shoppingList.items)
        ListTile(
          onTap: () {
            _toggleBought(item);
          },
          leading: Text('${item.quantity}x'),
          title: Text(item.name),
          trailing: Checkbox(
            value: item.isBought,
            onChanged: (_) {
              _toggleBought(item);
            },
          ),
        )
    ],
  );
}
```

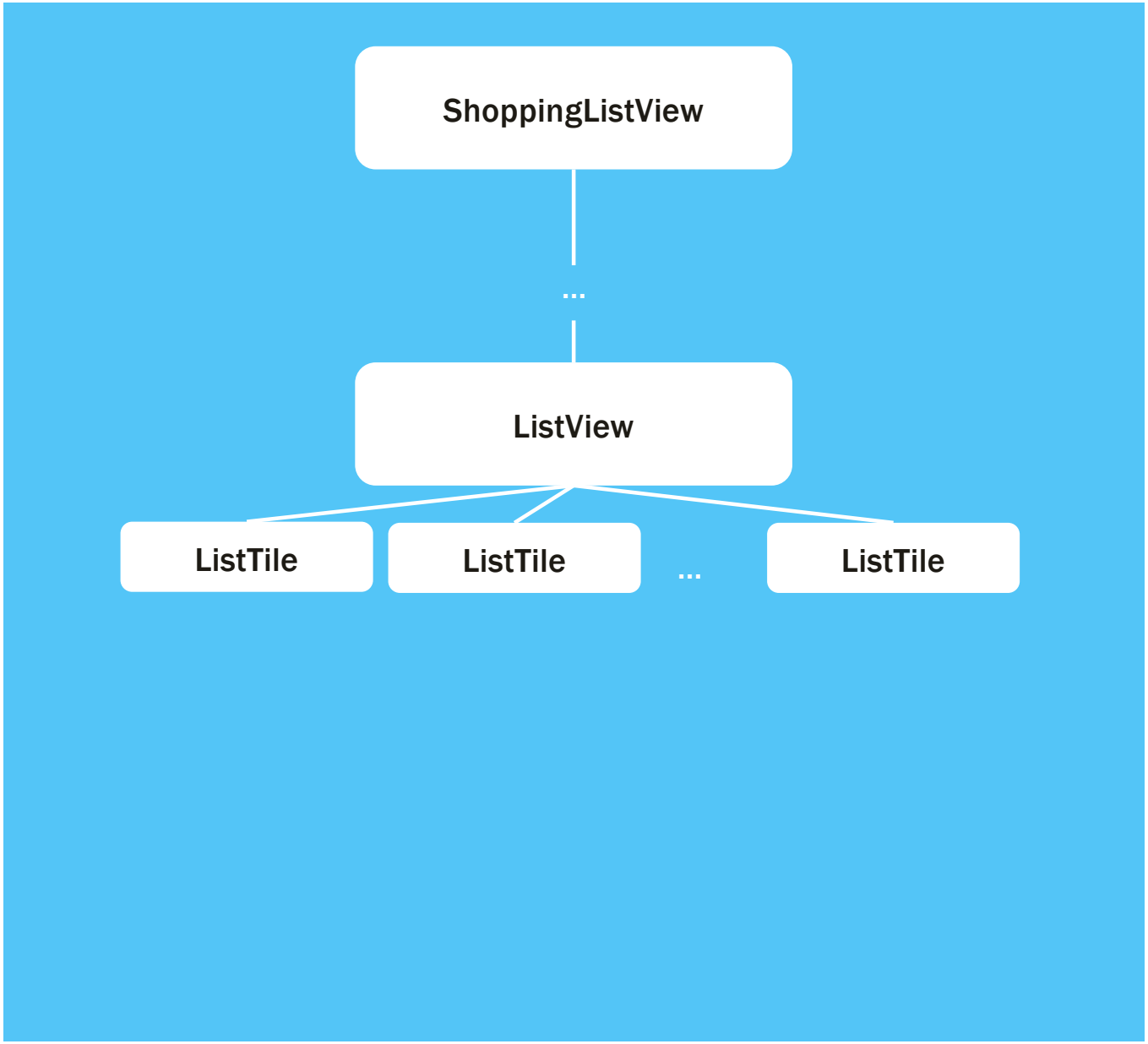
👆 renders all items at once

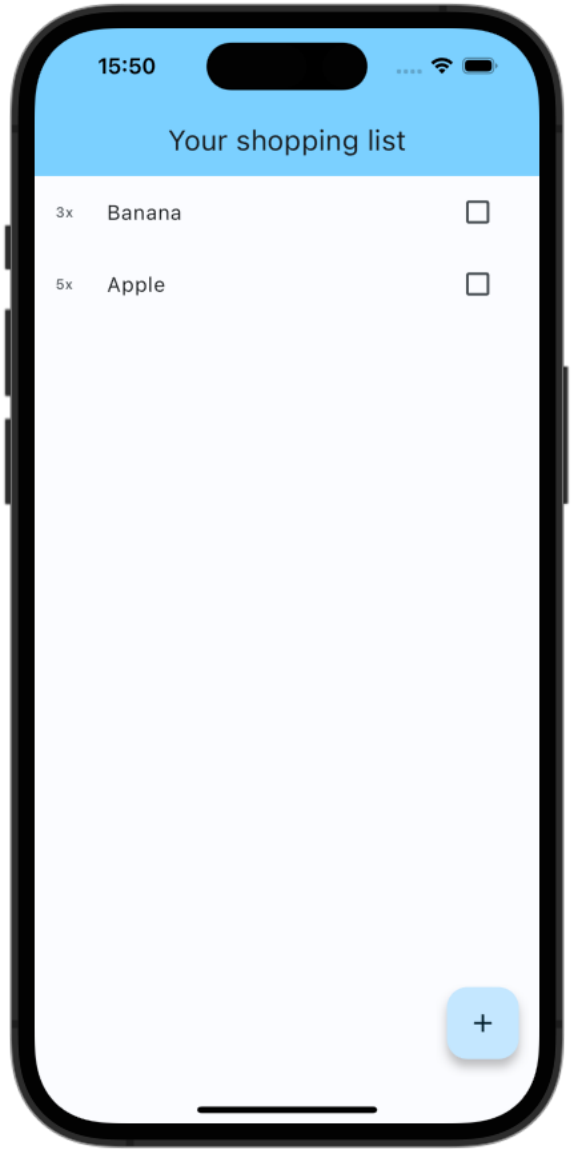
👆 rerenders the whole widget sub tree

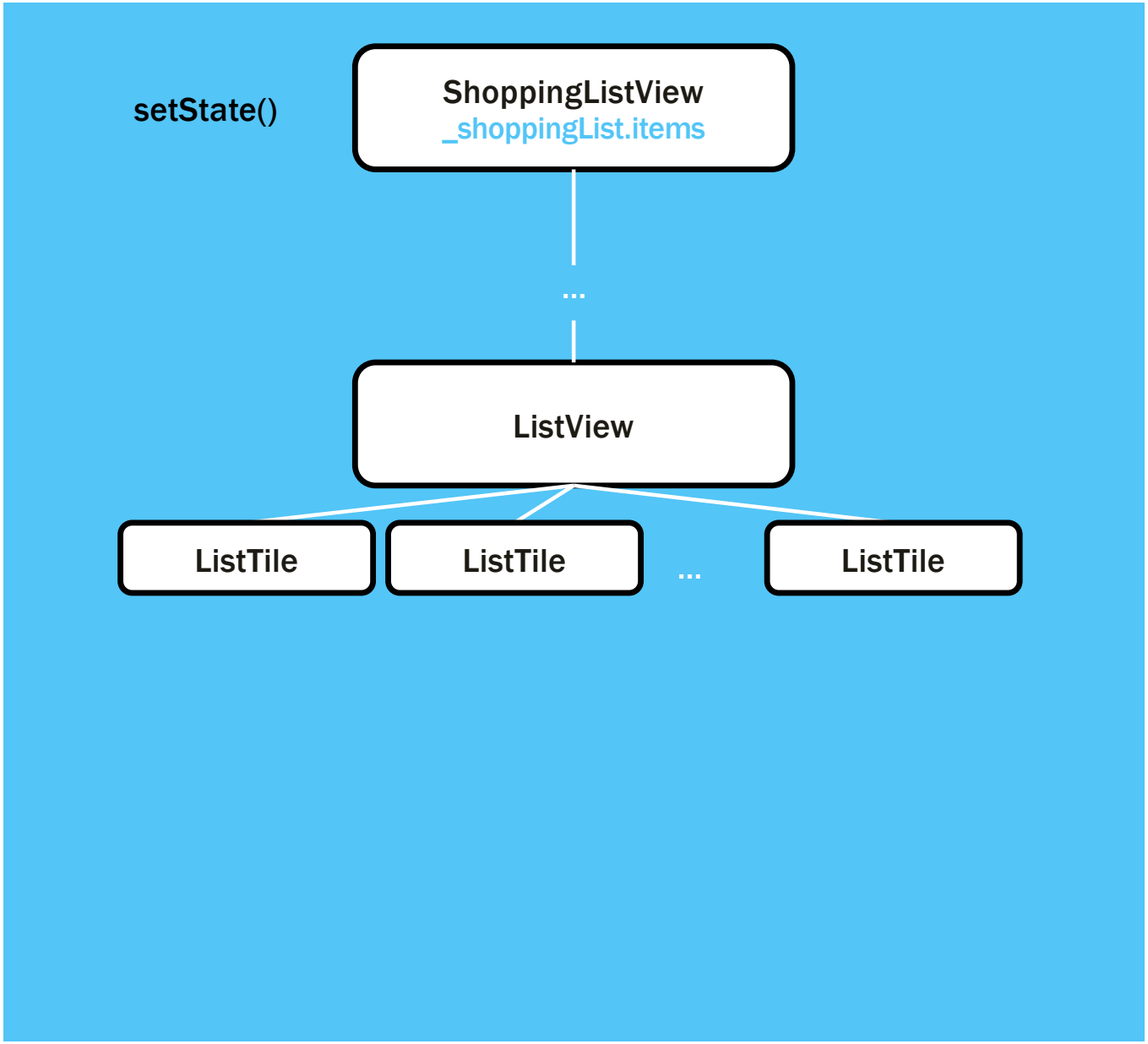
Why does setState rerender every ListTile?

Shopping List App


15:50
Your shopping list
3x Banana
5x Apple
+

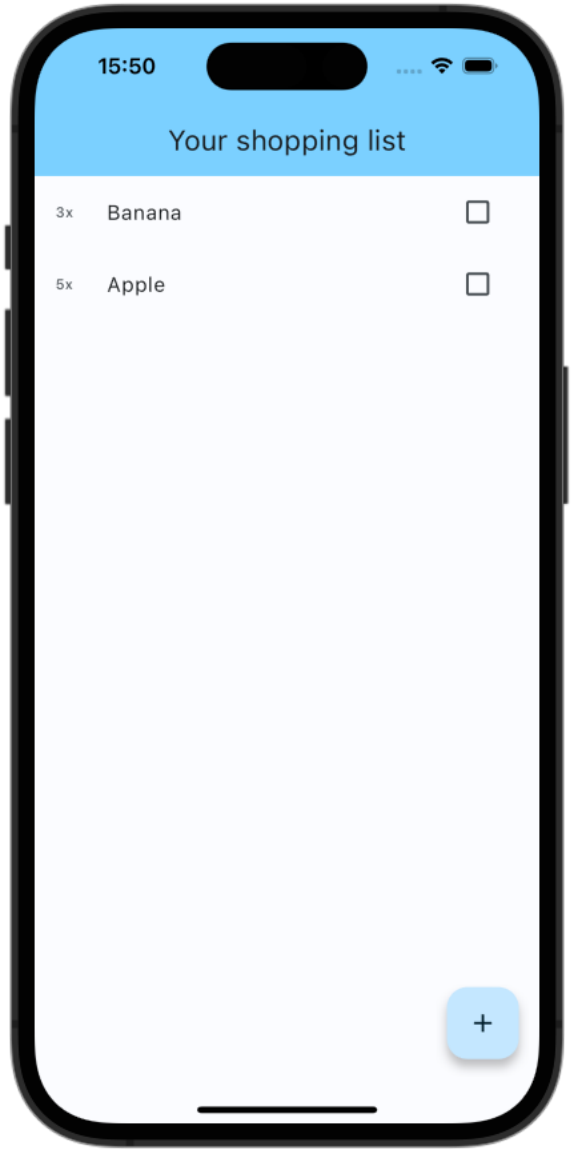

ShoppingListView
...
ListView
ListTile
ListTile
...
ListTile

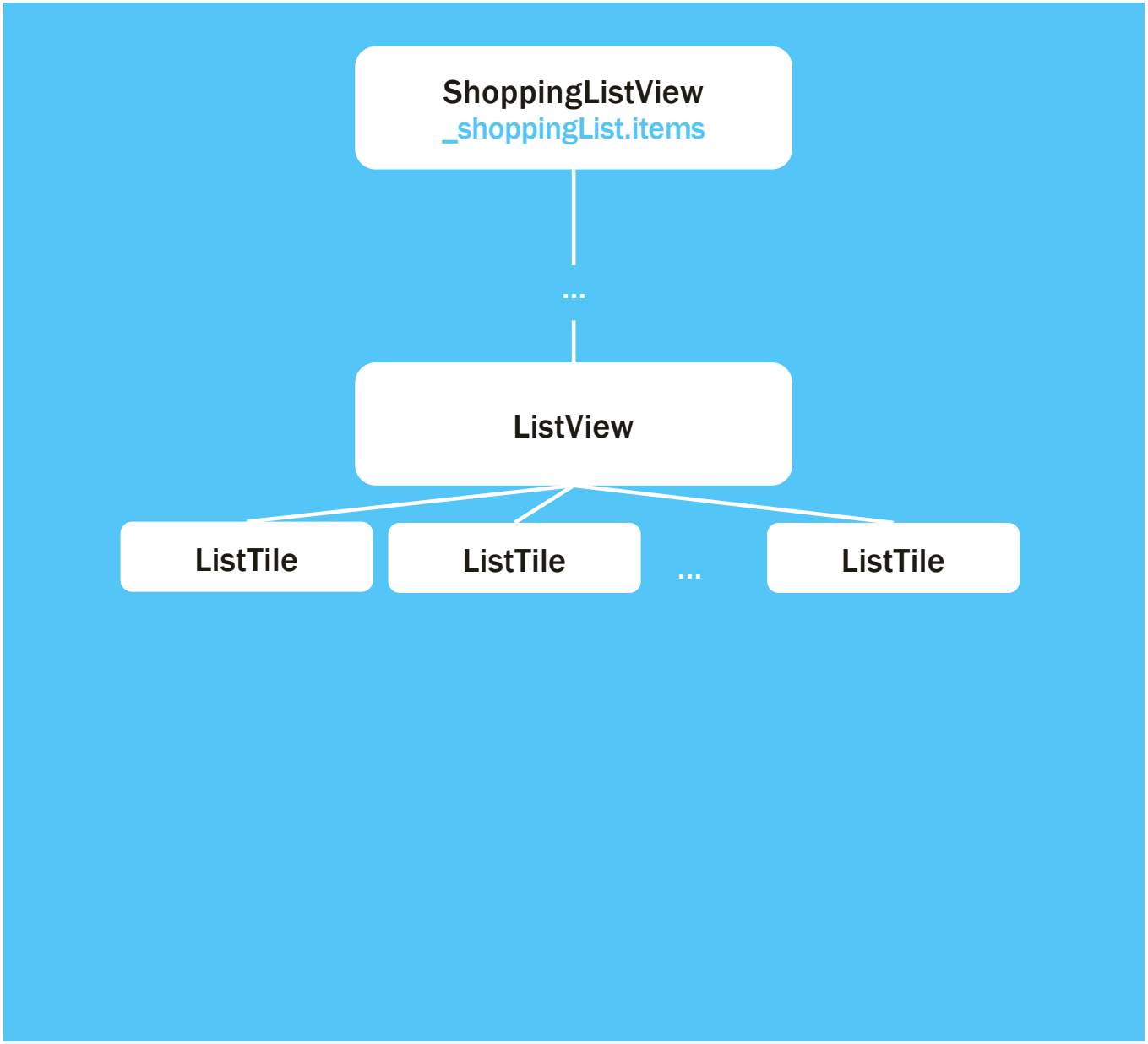

15:50
Your shopping list
3x Banana
5x Apple
+

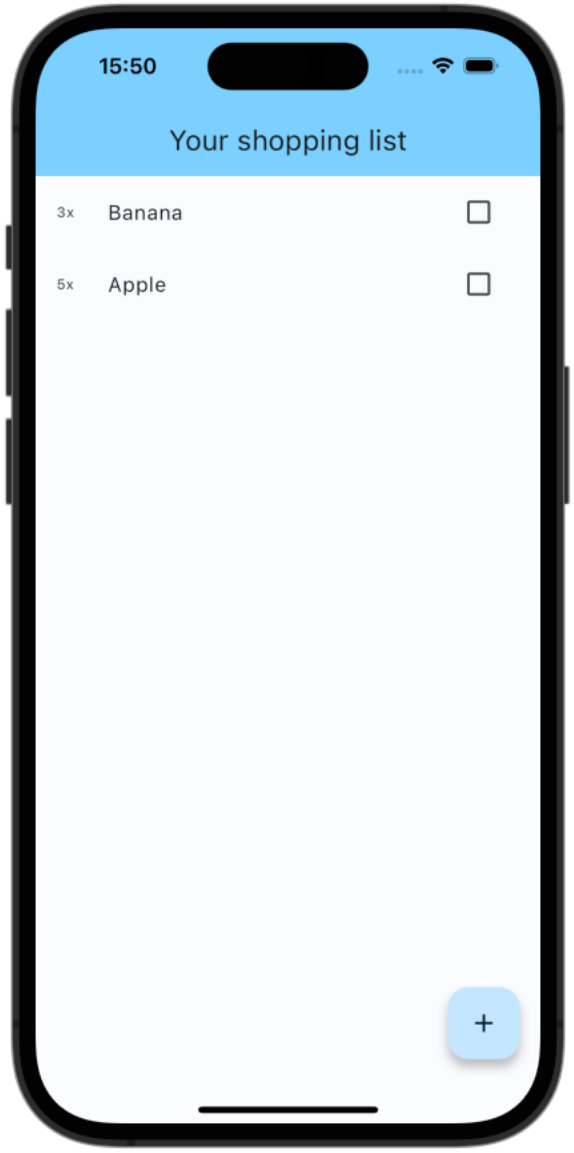

setState()
ShoppingListView
_shoppingList.items
...
ListView
ListTile
ListTile
...
ListTile

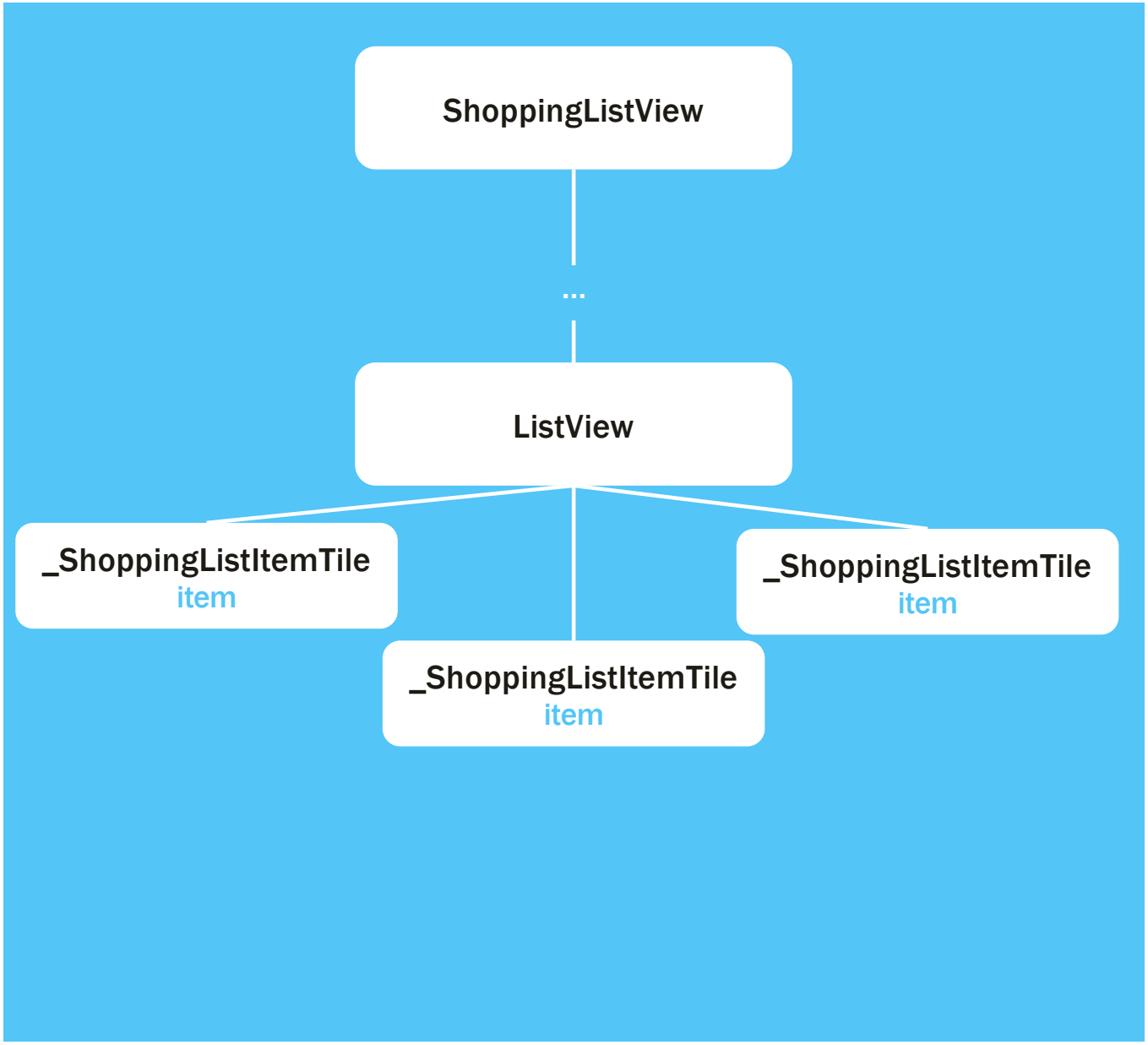
Lowering state

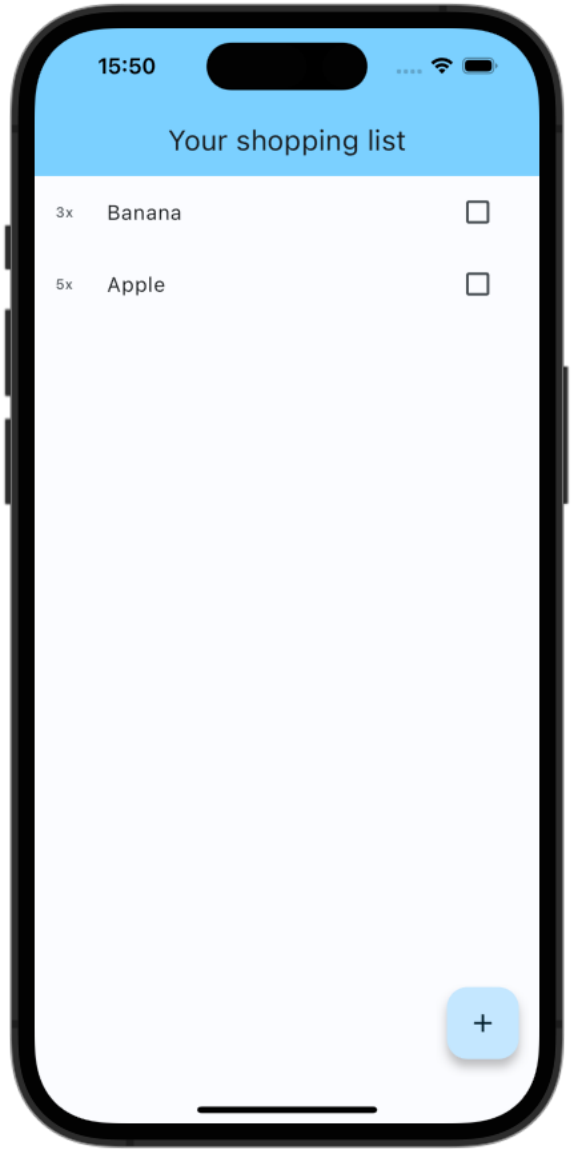
Shopping List App

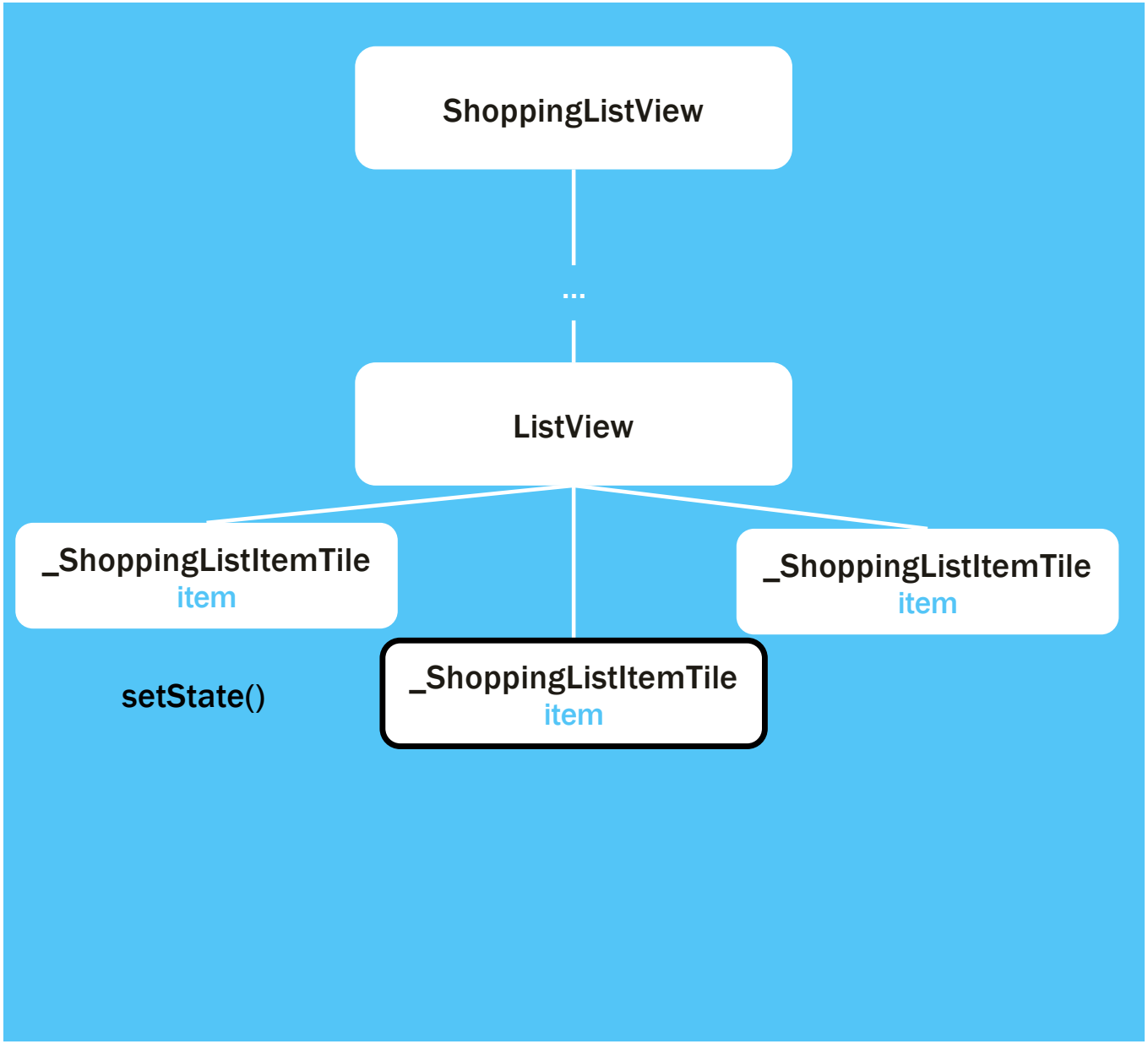

15:50
Your shopping list
3x Banana
5x Apple
+


ShoppingListView
_shoppingList.items
...
ListView
ListTile
ListTile
...
ListTile


15:50
Your shopping list
3x Banana
5x Apple
+

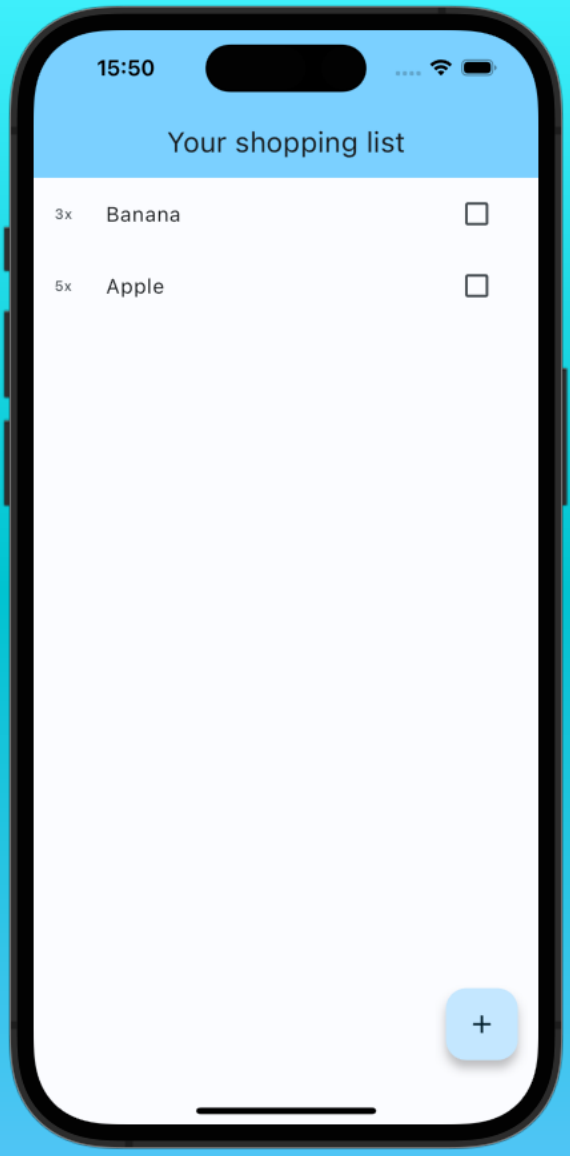

ShoppingListView
...
ListView
_ShoppingListItemTile
item
_ShoppingListItemTile
item
_ShoppingListItemTile
item


15:50
Your shopping list
3x Banana
5x Apple
+

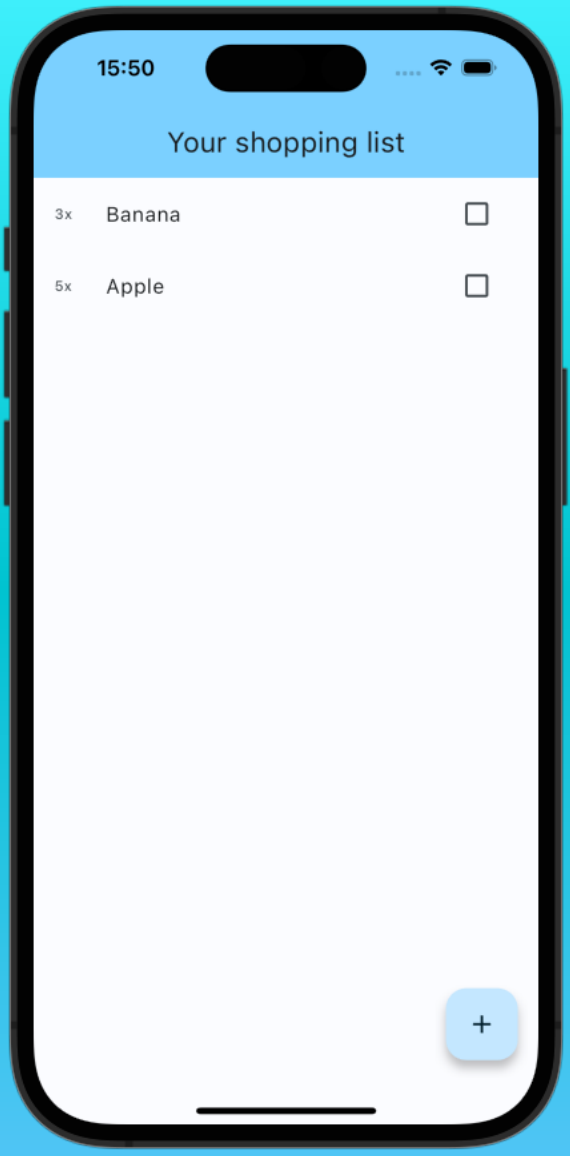

ShoppingListView
...
ListView
_ShoppingListItemTile
item
_ShoppingListItemTile
item
setState()
_ShoppingListItemTile
item

Implementation

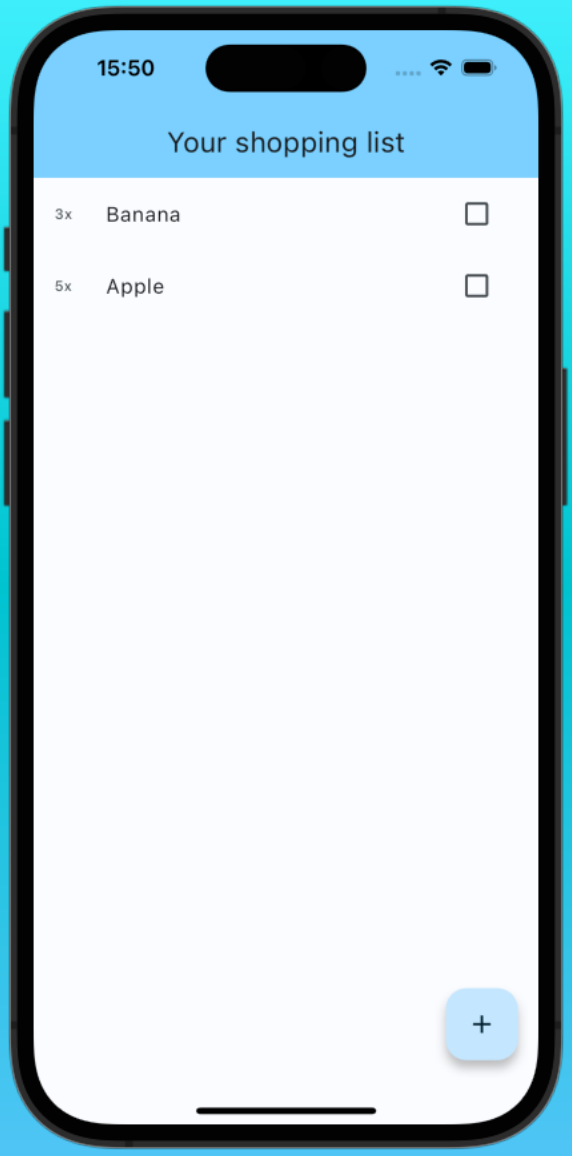
Shopping List App



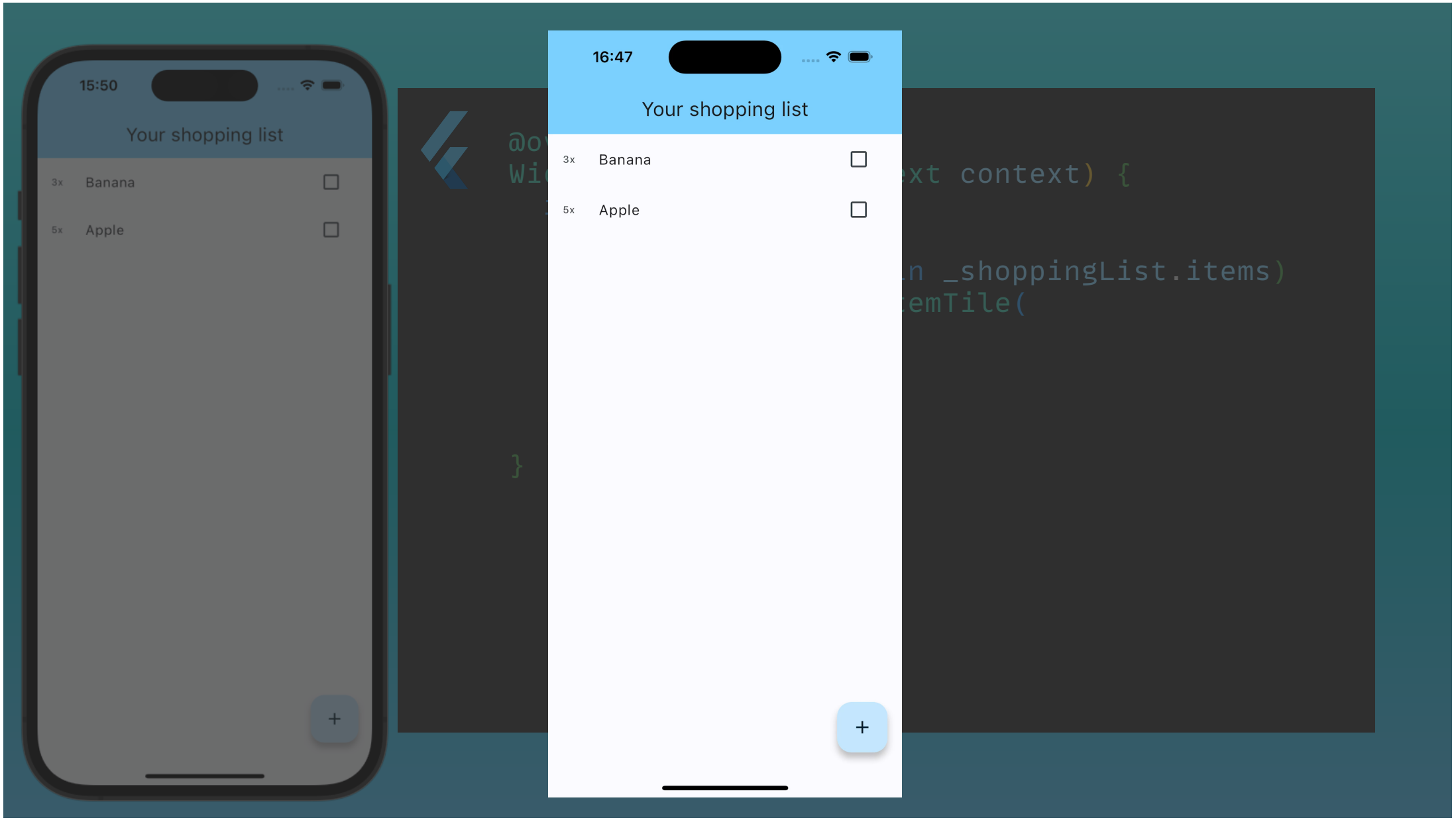

```dart
class _ShoppingListItemTile extends StatefulWidget {
  const _ShoppingListItemTile({
    required this.item,
    super.key,
  });

  final ShoppingListItem item;

  @override
  State<_ShoppingListItemTile> createState() =>
_ShoppingListItemTileState();
}


class _ShoppingListItemTileState extends
State<_ShoppingListItemTile> {
  @override
  Widget build(BuildContext context) {
    return const Placeholder();
  }
}
```




```dart
ShoppingListItem get item => widget.item;

void _toggleBought() {
  setState(() {
    item.isBought = !item.isBought;
  });
}

@override
Widget build(BuildContext context) {
  return ListTile(
    onTap: _toggleBought,
    leading: Text('${item.quantity}x'),
    title: Text(item.name),
    trailing: Checkbox(
      value: item.isBought,
      onChanged: (_) {
        _toggleBought();
      },
    ),
  );
}
```

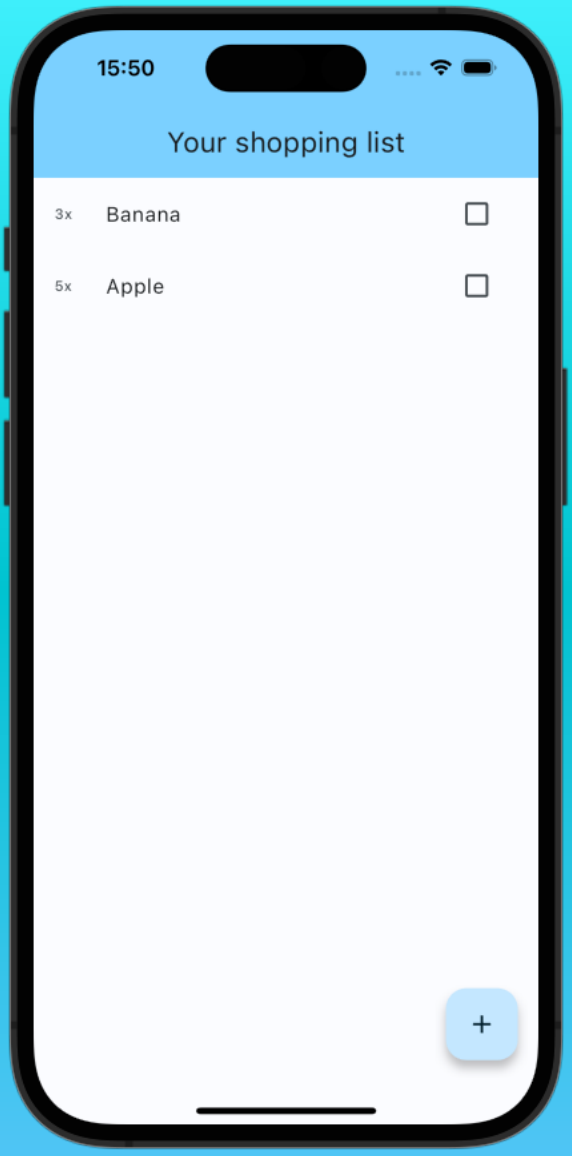



```dart
@override
Widget build(BuildContext context) {
  return ListView(
    children: [
      for (final item in _shoppingList.items)
        _ShoppingListItemTile(
          item: item,
        ),
    ],
  );
}
```


15:50
Your shopping list
3x Banana
5x Apple
+
16:47
Your shopping list
3x Banana
5x Apple
+
@o
Wi
ext context) {
n _shoppingList.items)
emTile(
}

Lazy Rendering with ListView.builder

Shopping List App




```dart
@override
Widget build(BuildContext context) {
  return ListView(
    children: [
      for (final item in _shoppingList.items)
        _ShoppingListItemTile(
          item: item,
        ),
    ],
  );
}
```

👆 renders all ListTiles before rendering the ListView

ListView.builder

”Creates a scrollable, linear array of widgets that are created on demand.“

ListView.builder

- itemCount: Amount of widgets to render.
- itemBuilder: Callback that defines the rendered widget.
- see all properties here

Further ListViews

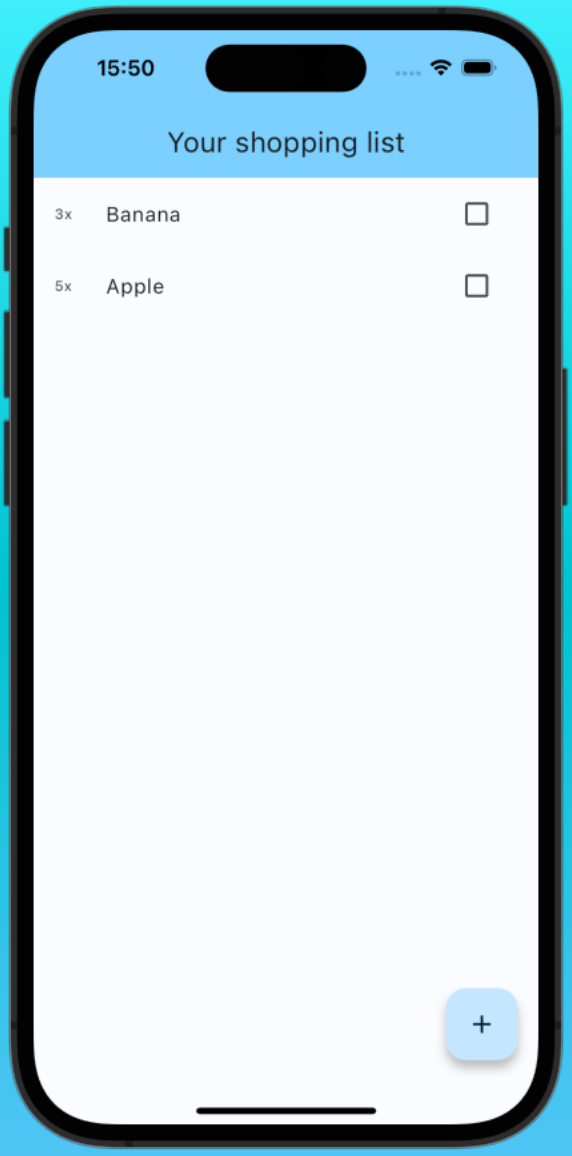
- ListView.separated: “Creates a fixed-length scrollable linear array of list “items” separated by list item “separators”.”

Further ListViews

- ListView.separated: “Creates a fixed-length scrollable linear array of list “items” separated by list item “separators”.”
- ListView.custom: “Creates a scrollable, linear array of widgets with a custom child model.”

Using ListView.builder

Shopping List App

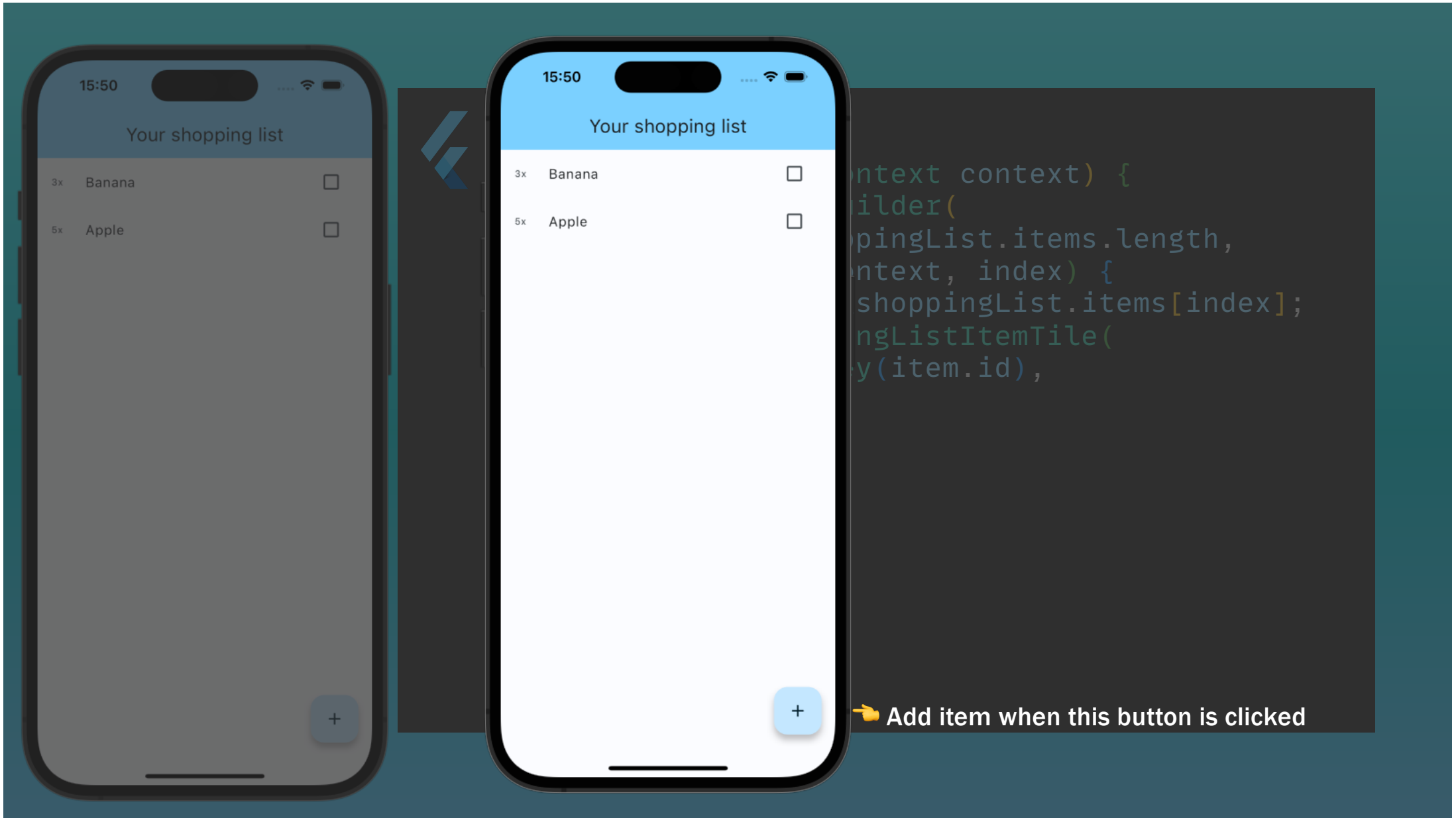


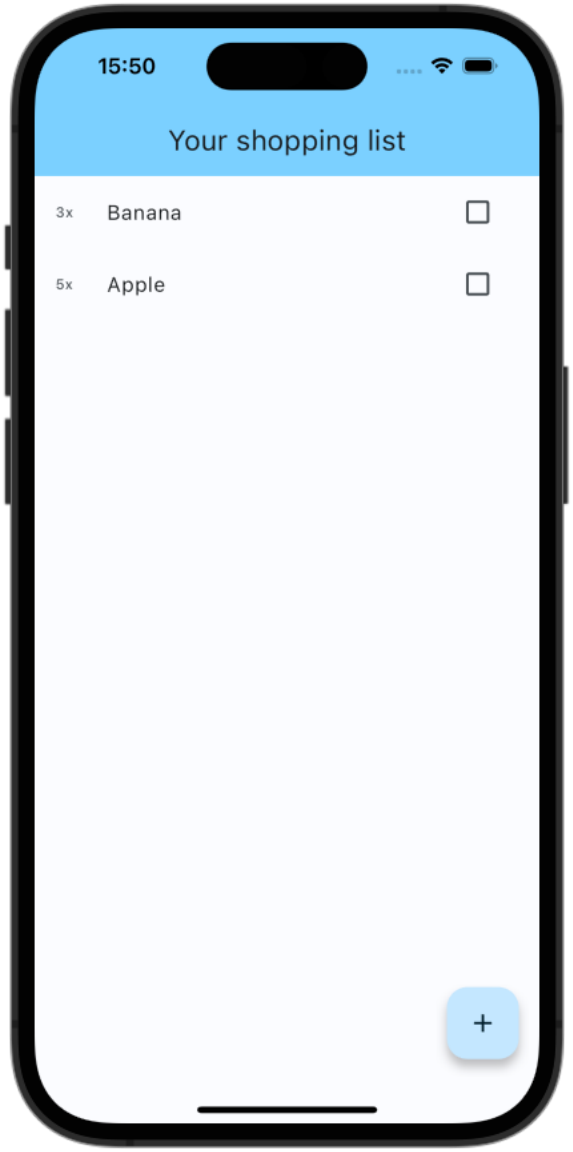

```dart
@override
Widget build(BuildContext context) {
  return ListView.builder(
    itemCount: _shoppingList.items.length,
    itemBuilder: (context, index) {
      final item = _shoppingList.items[index];
      return _ShoppingListItemTile(
        key: ValueKey(item.id),
        item: item,
      );
    },
  );
}
```

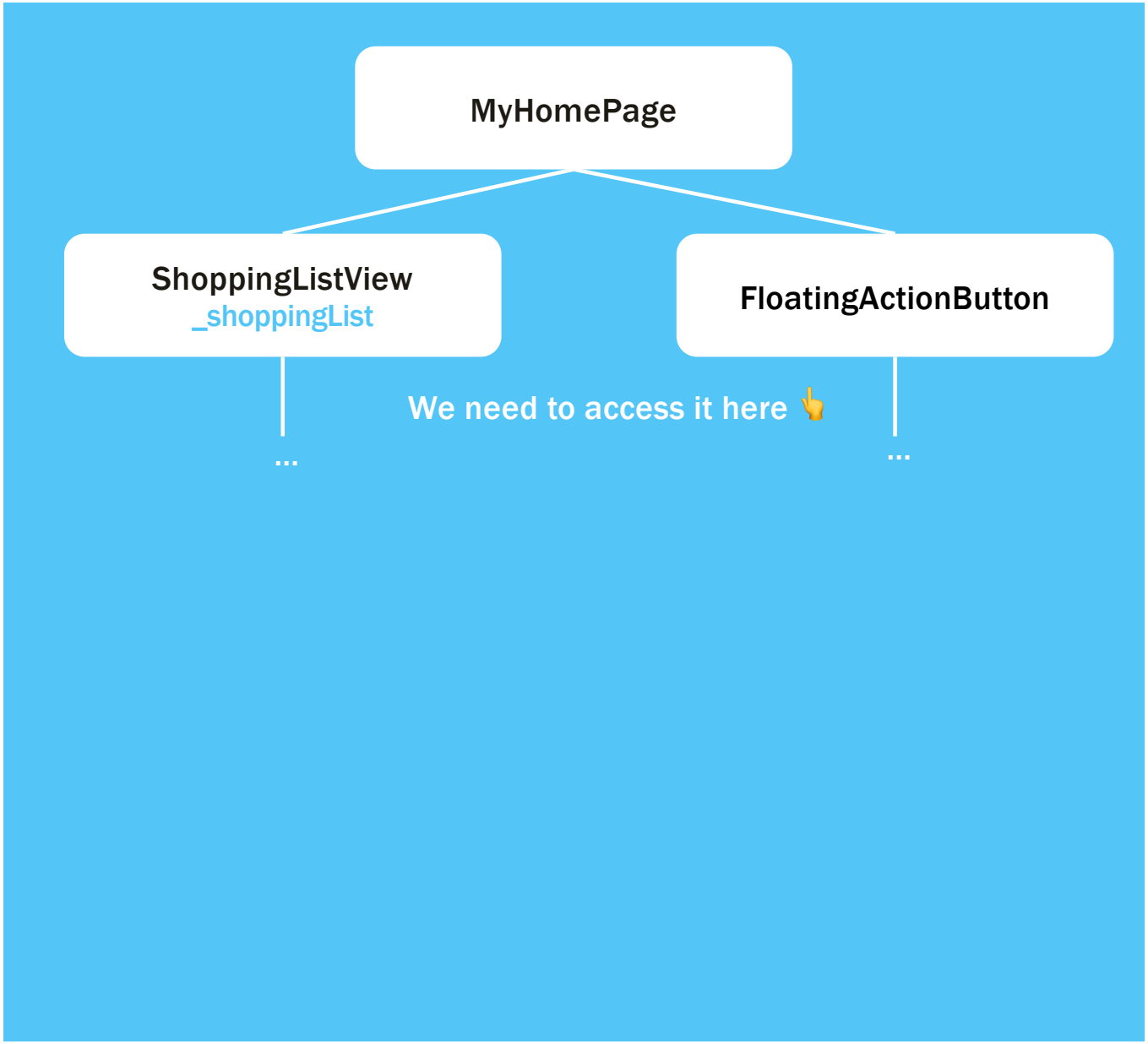
👆 use key to uniquely identify the widget

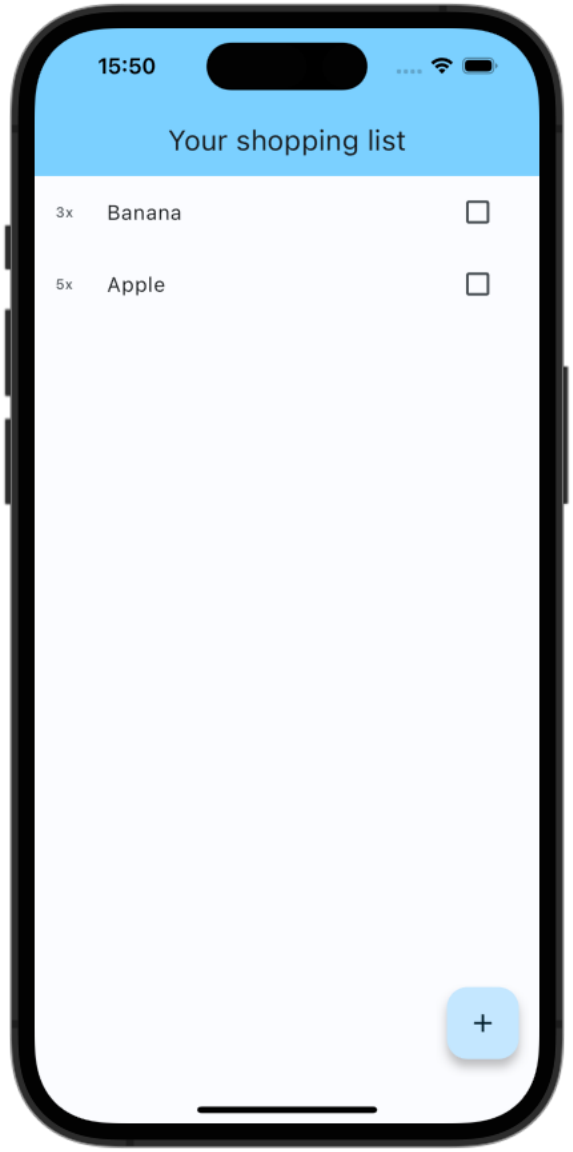
Adding Items

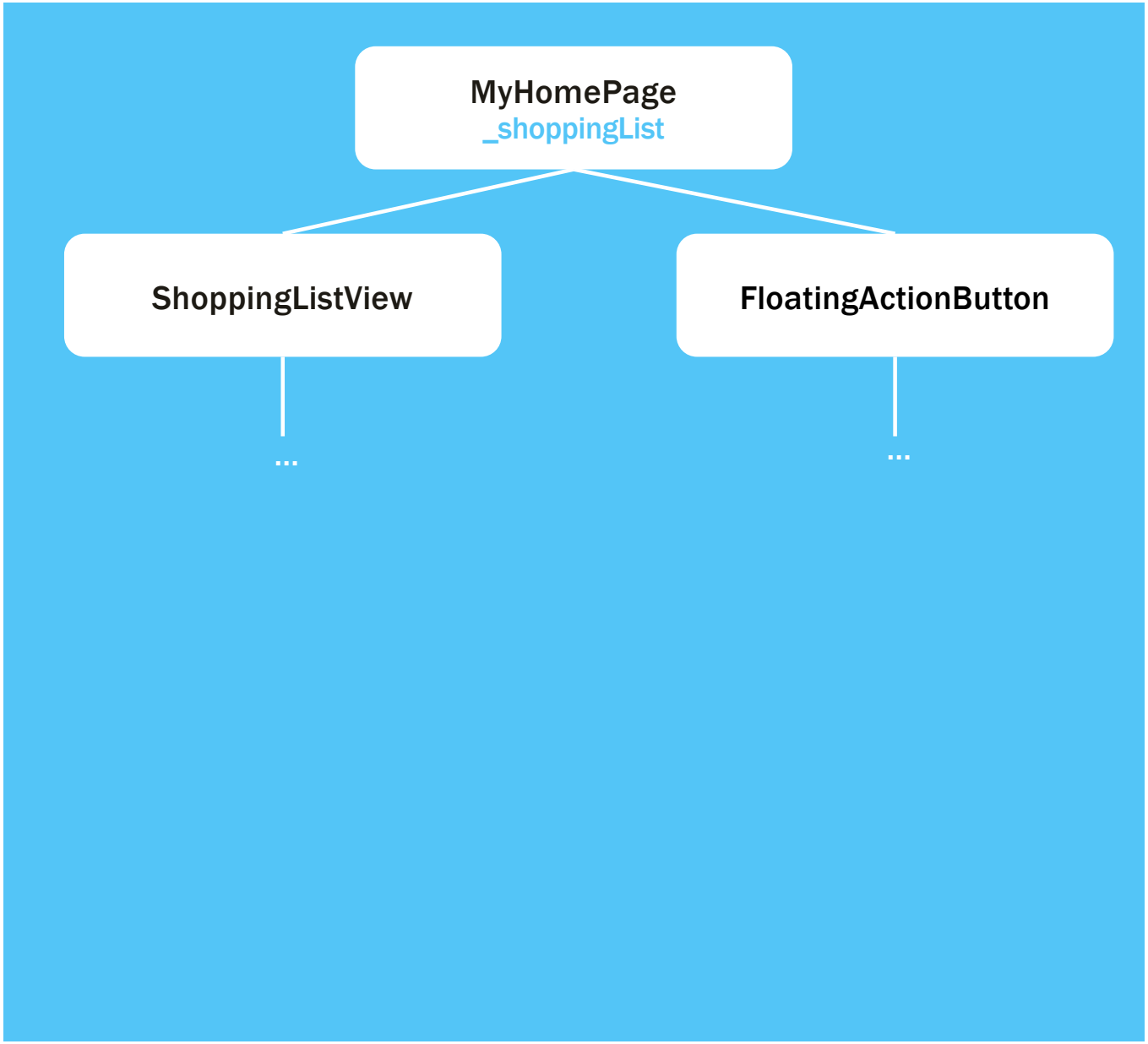
Shopping List App

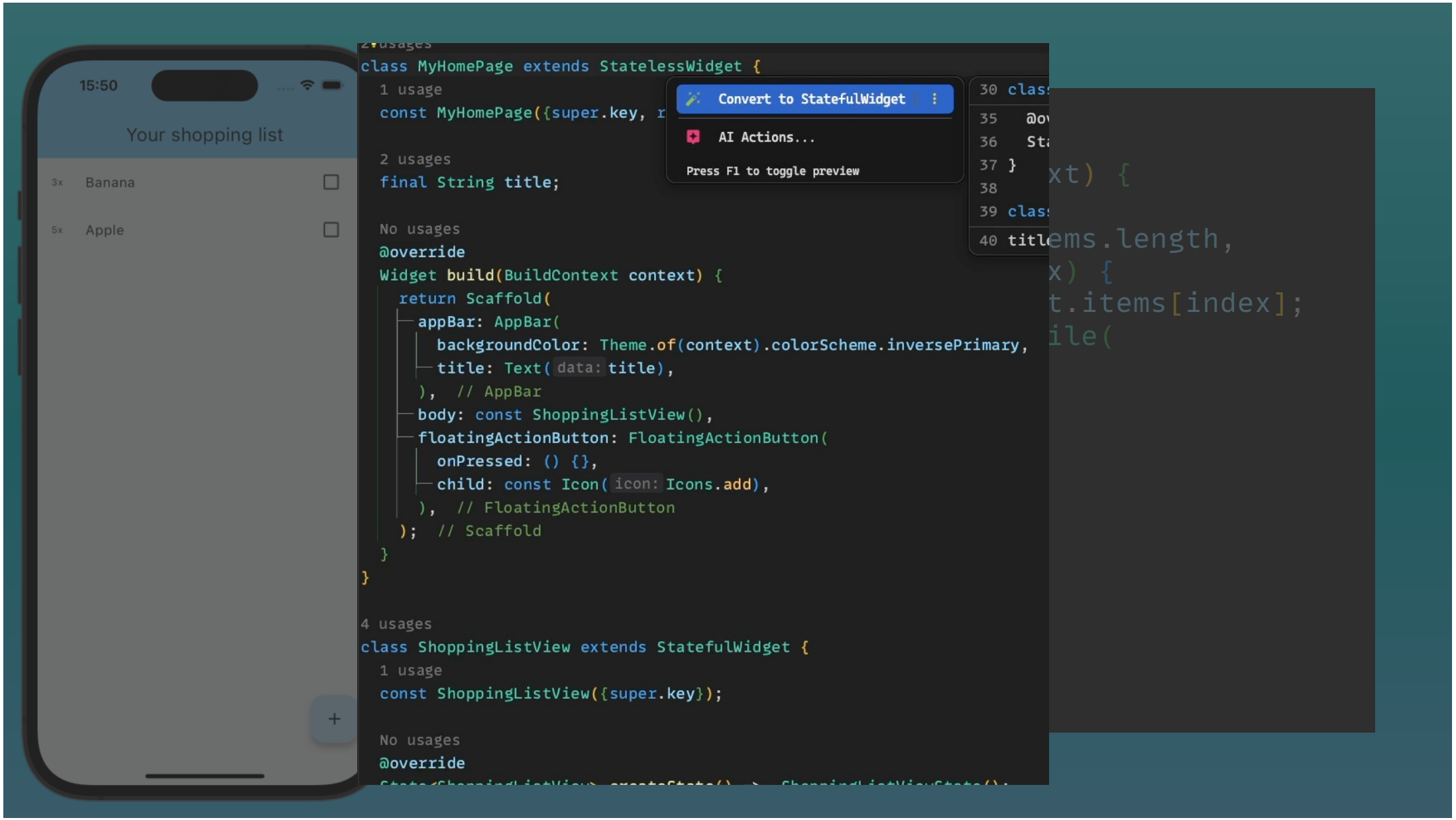

15:50
Your shopping list
3x Banana
5x Apple
+
15:50
Your shopping list
3x Banana
5x Apple
+
ntext context) {
ilder(
pingList.items.length,
ntext, index) {
shoppingList.items[index];
ngListItemTile(
y(item.id),
Add item when this button is clicked

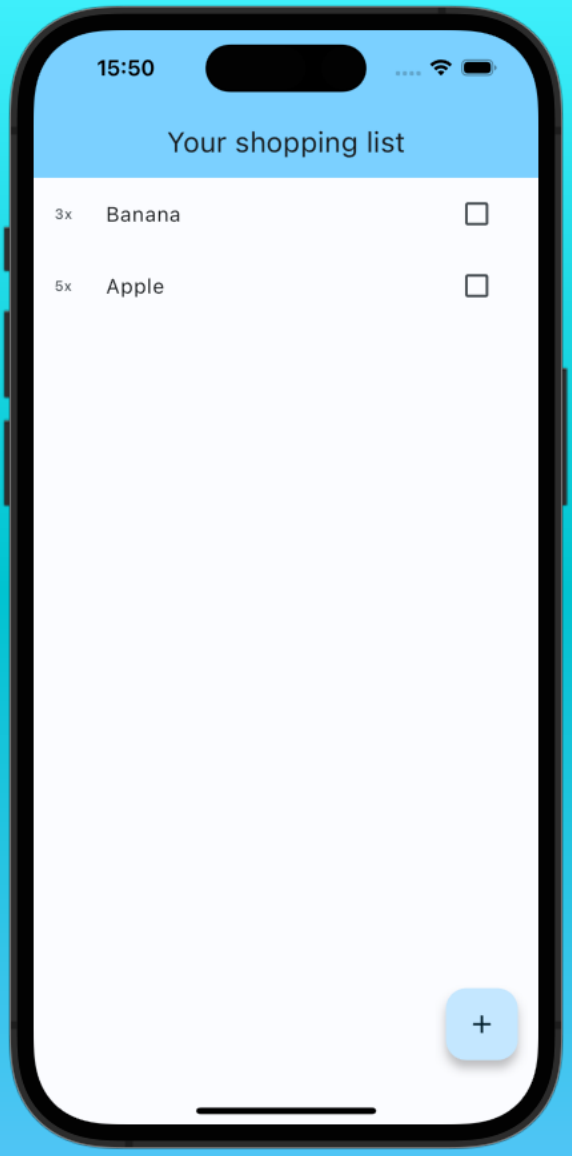

15:50
Your shopping list
3x Banana
5x Apple
+


MyHomePage
ShoppingListView
_shoppingList
FloatingActionButton
We need to access it here 👆
...
...

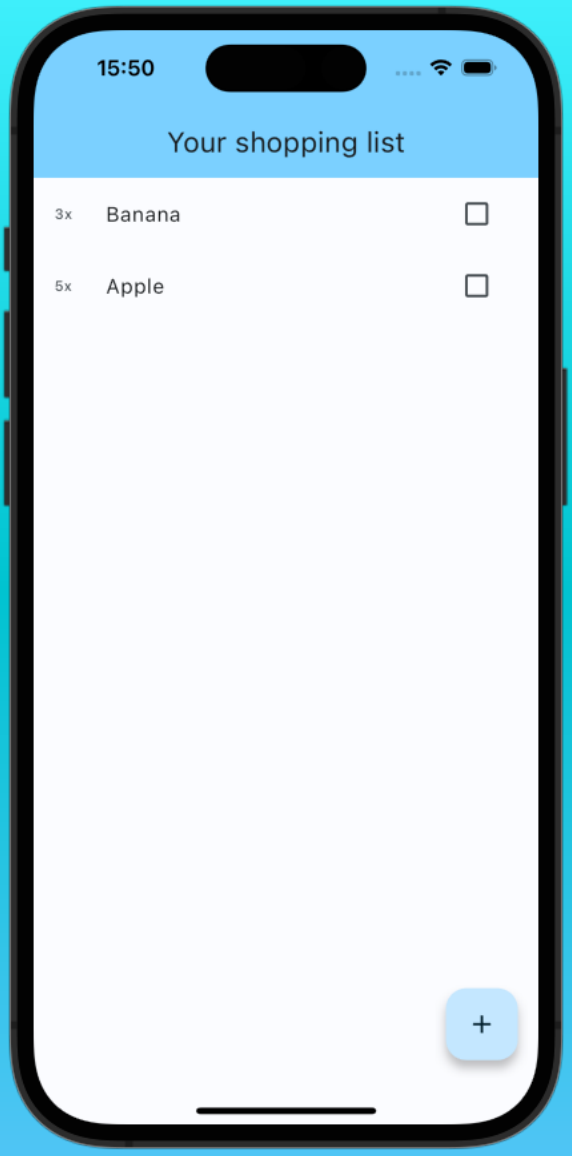

15:50
Your shopping list
3x Banana
5x Apple
+


MyHomePage
_shoppingList
ShoppingListView
FloatingActionButton
...
...

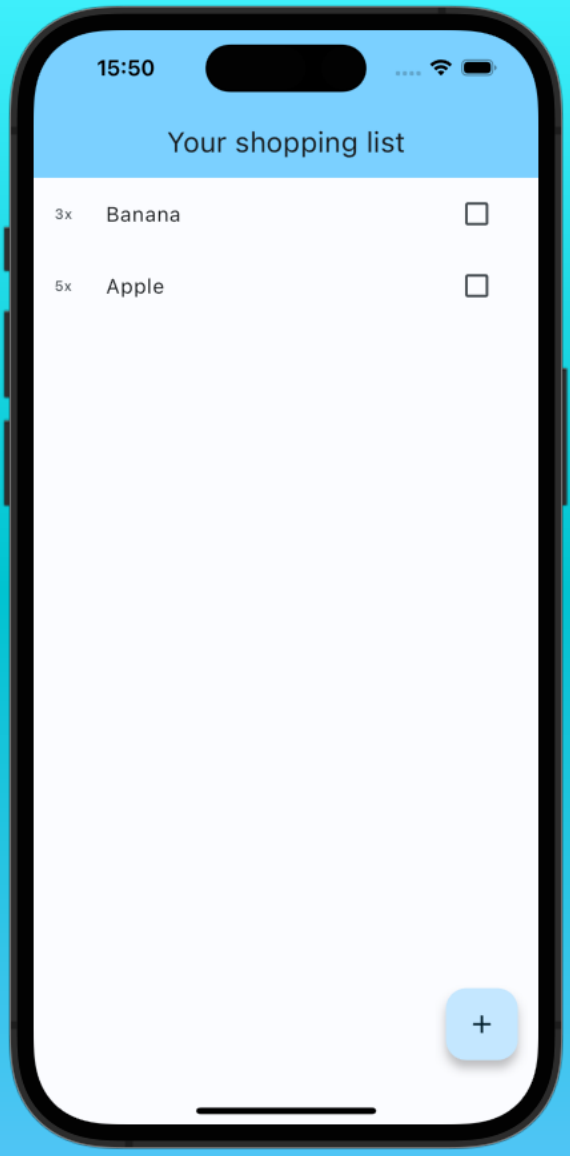

15:50
Your shopping list
3x Banana
5x Apple
class MyHomePage extends StatelessWidget {
1 usage
const MyHomePage({super.key, r
2 usages
final String title;
No usages
@override
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
backgroundColor: Theme.of(context).colorScheme.inversePrimary,
title: Text(data: title),
), // AppBar
body: const ShoppingListView(),
floatingActionButton: FloatingActionButton(
onPressed: () {},
child: const Icon(icon: Icons.add),
), // FloatingActionButton
); // Scaffold
}
}
4 usages
class ShoppingListView extends StatefulWidget {
1 usage
const ShoppingListView({super.key});
No usages
@override
Convert to StatefulWidget
AI Actions...
Press F1 to toggle preview
30 class
35 @ov
36 Sta
37 }
38
39 class
40 title



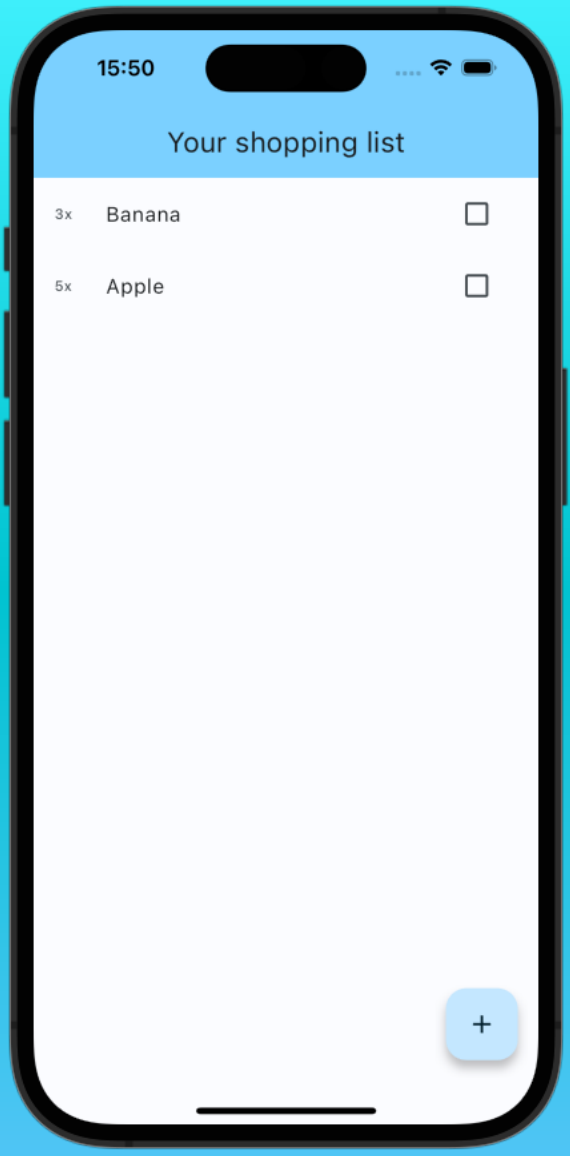

```dart
class _MyHomePageState extends State<MyHomePage>
{

  ShoppingList _shoppingList = ShoppingList()
    ..addItem(
      name: 'Banana',
      quantity: 3,
    )
    ..addItem(
      name: 'Apple',
      quantity: 5,
    );

  …
}
```



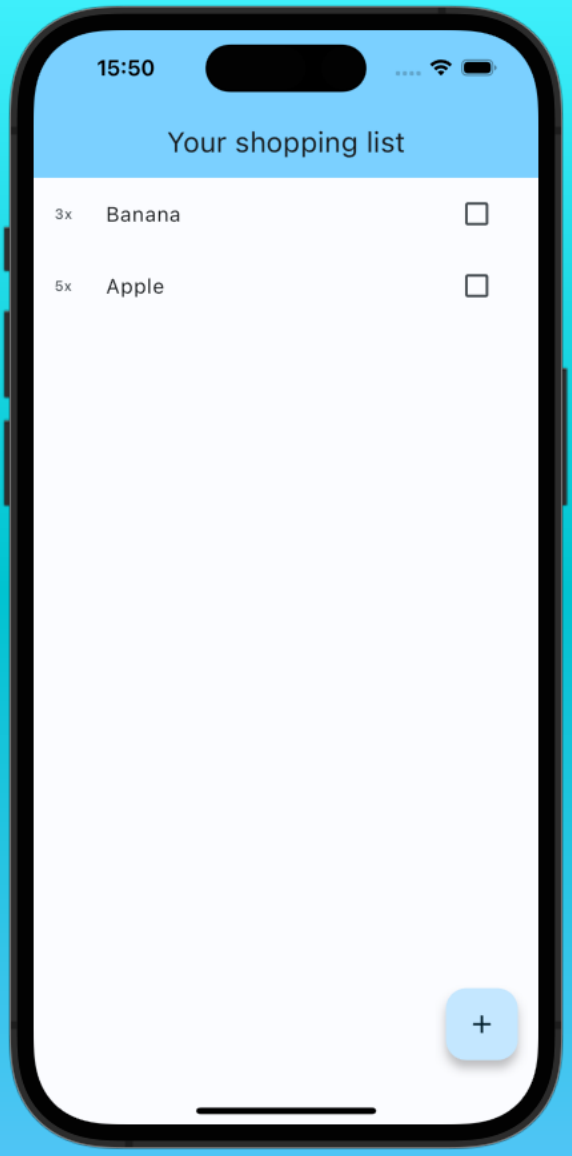

```dart
class ShoppingListView extends StatefulWidget {
  const ShoppingListView({super.key});

  @override
  State<ShoppingListView> createState() =>
_ShoppingListViewState();
}

class _ShoppingListViewState extends State<ShoppingListView>
{
  @override
  Widget build(BuildContext context) {
    return ListView.builder(
      itemCount: _shoppingList.items.length,
      itemBuilder: (context, index) {
        final item = _shoppingList.items[index];
        return _ShoppingListItemTile(
          key: ValueKey(item.id),
          item: item,
        );
      },
    );
  }
}
```



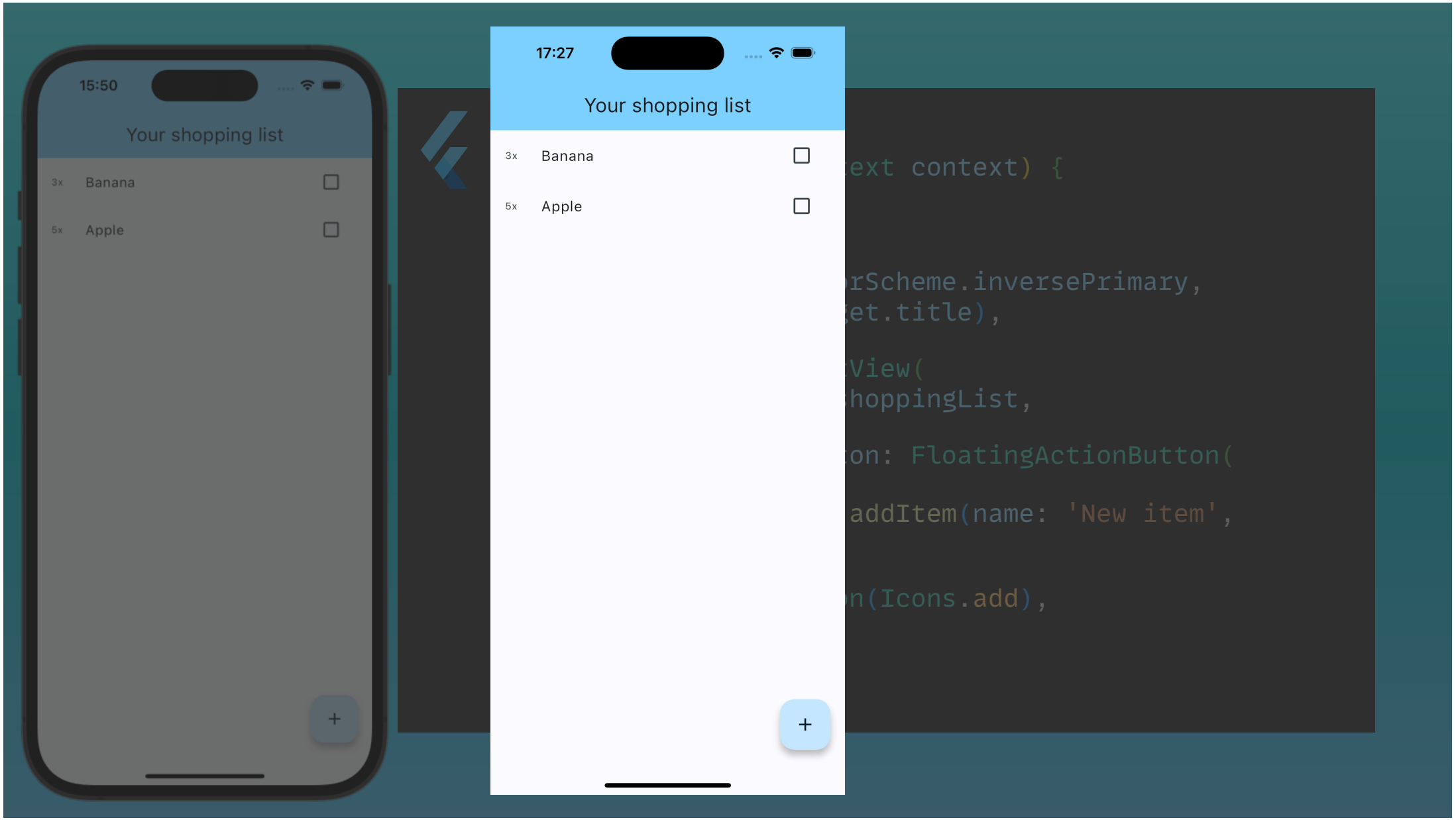

```dart
class ShoppingListView extends StatelessWidget {
  const ShoppingListView({super.key});

  @override
  Widget build(BuildContext context) {
    return ListView.builder(
      itemCount: _shoppingList.items.length,
      itemBuilder: (context, index) {
        final item = _shoppingList.items[index];
        return _ShoppingListItemTile(
          key: ValueKey(item.id),
          item: item,
        );
      },
    );
  }
}
```




```dart
class ShoppingListView extends StatelessWidget {
  const ShoppingListView({
    required this.shoppingList,
    super.key,
  });

  final ShoppingList shoppingList;
  @override
  Widget build(BuildContext context) {
    return ListView.builder(
      itemCount: shoppingList.items.length,
      itemBuilder: (context, index) {
        final item = shoppingList.items[index];
        return _ShoppingListItemTile(
          key: ValueKey(item.id),
          item: item,
        );
      },
    );
  }
}
```




```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
      title: Text(widget.title),
    ),
    body: ShoppingListView(
      shoppingList: _shoppingList,),
    floatingActionButton: FloatingActionButton(
      onPressed: () {
        setState(() {
          _shoppingList.addItem(name: 'New item',
quantity: 1);
        });
      },
      child: const Icon(Icons.add),
    ),
  );
}
```

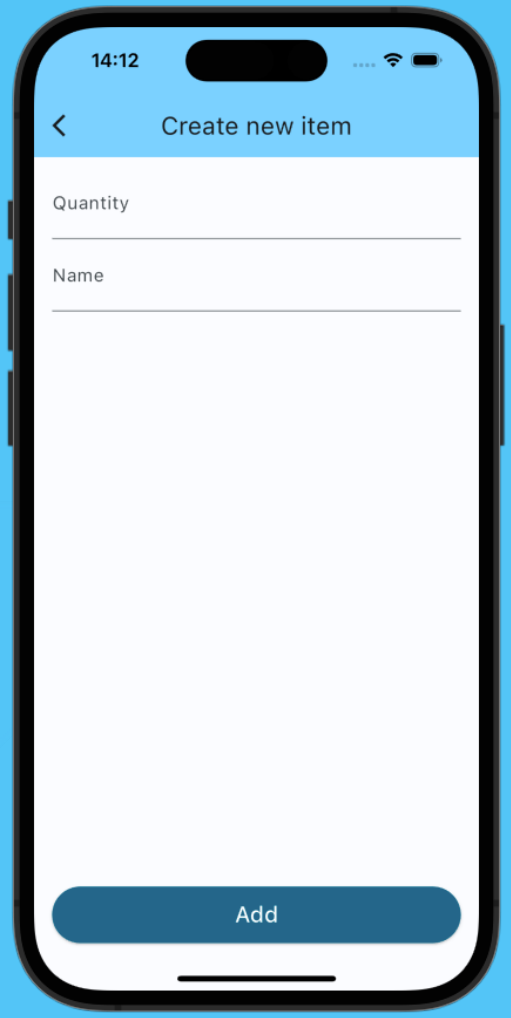

15:50
Your shopping list
3x Banana
5x Apple
17:27
Your shopping list
3x Banana
5x Apple
ext context) {
rScheme.inversePrimary,
et.title),
View(
hoppingList,
on: FloatingActionButton(
addItem(name: 'New item',
n(Icons.add),

Creating a new page to add items

Shopping List App

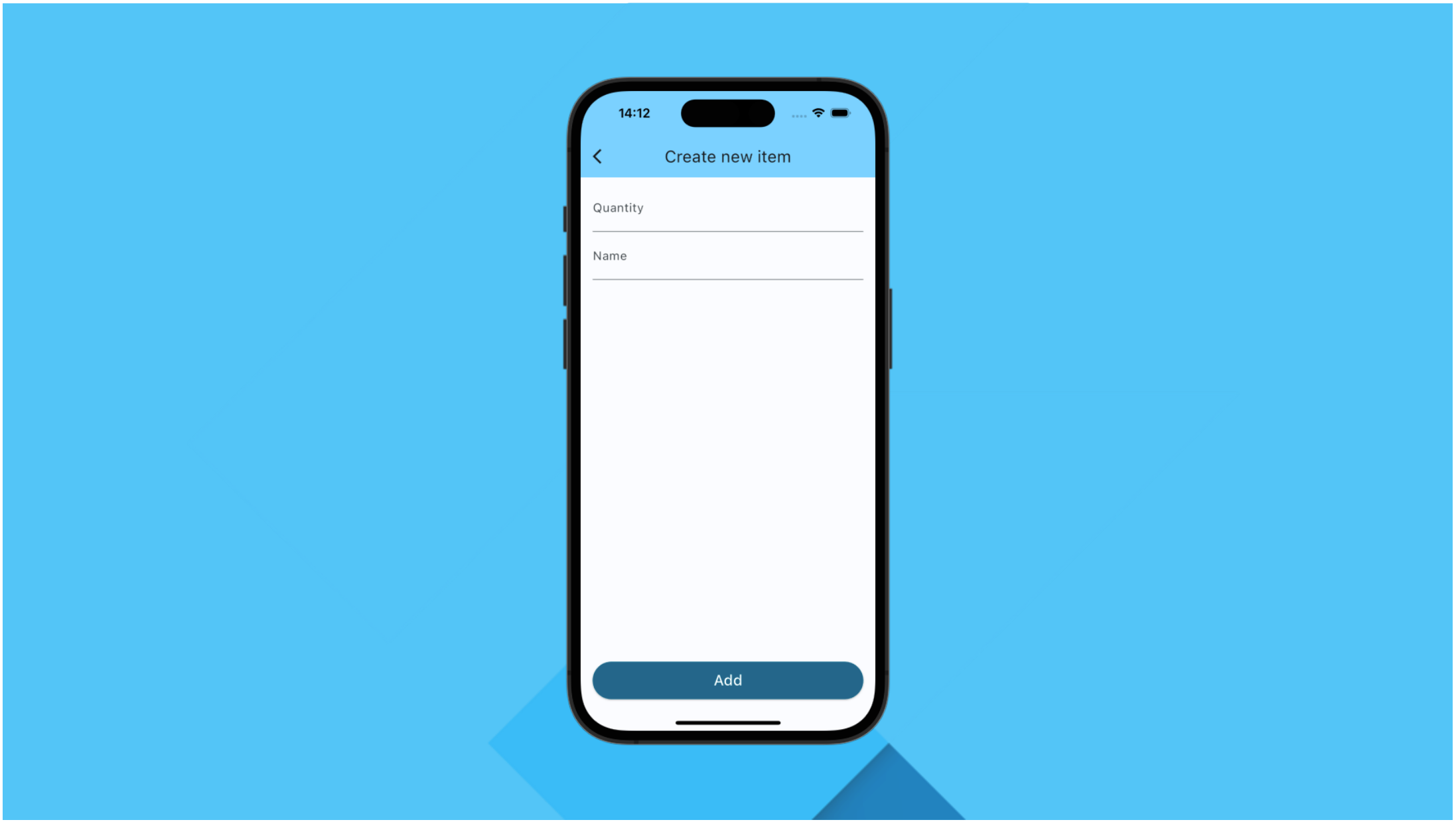
Creating a new page to add items

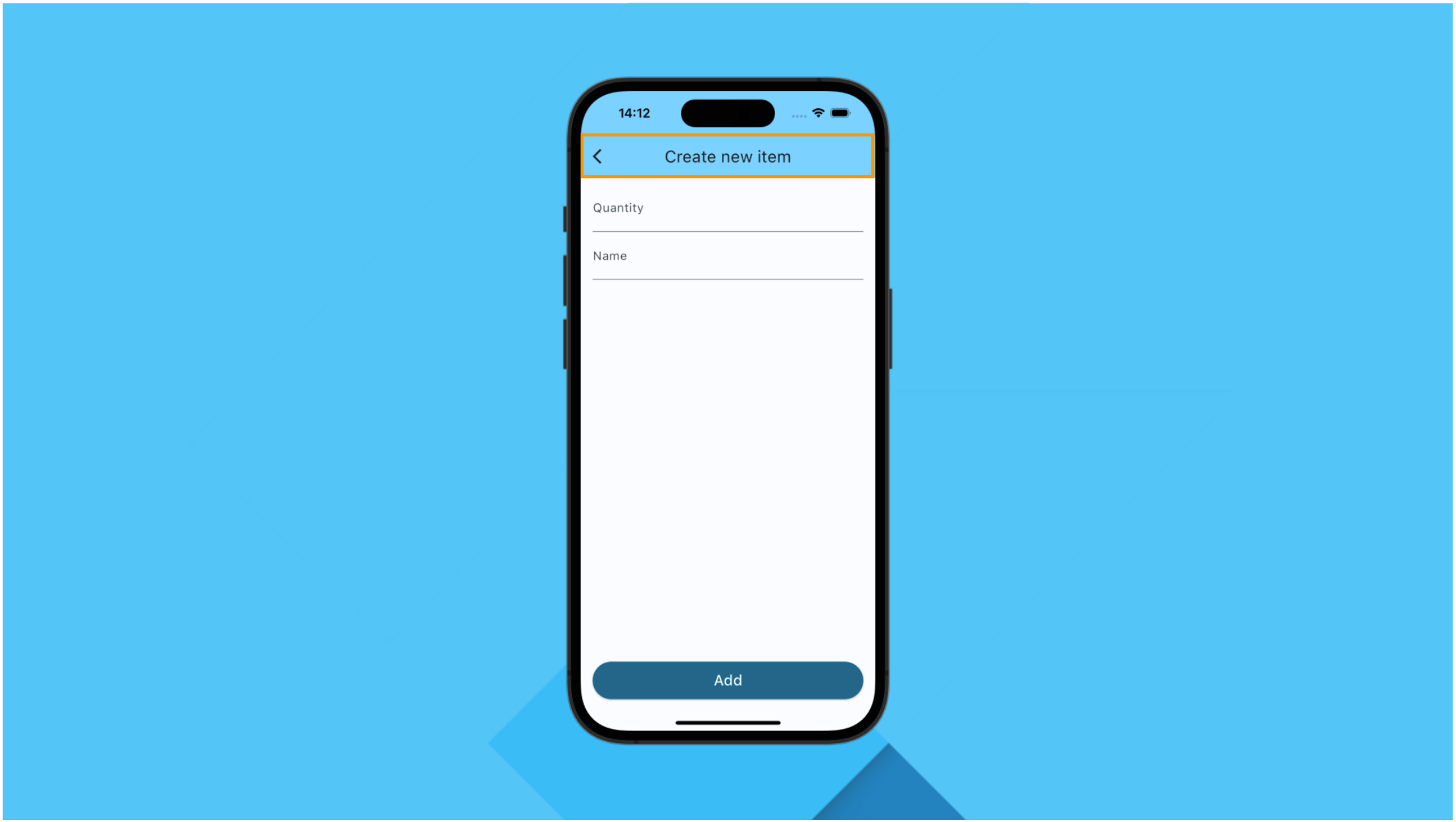
Shopping List App

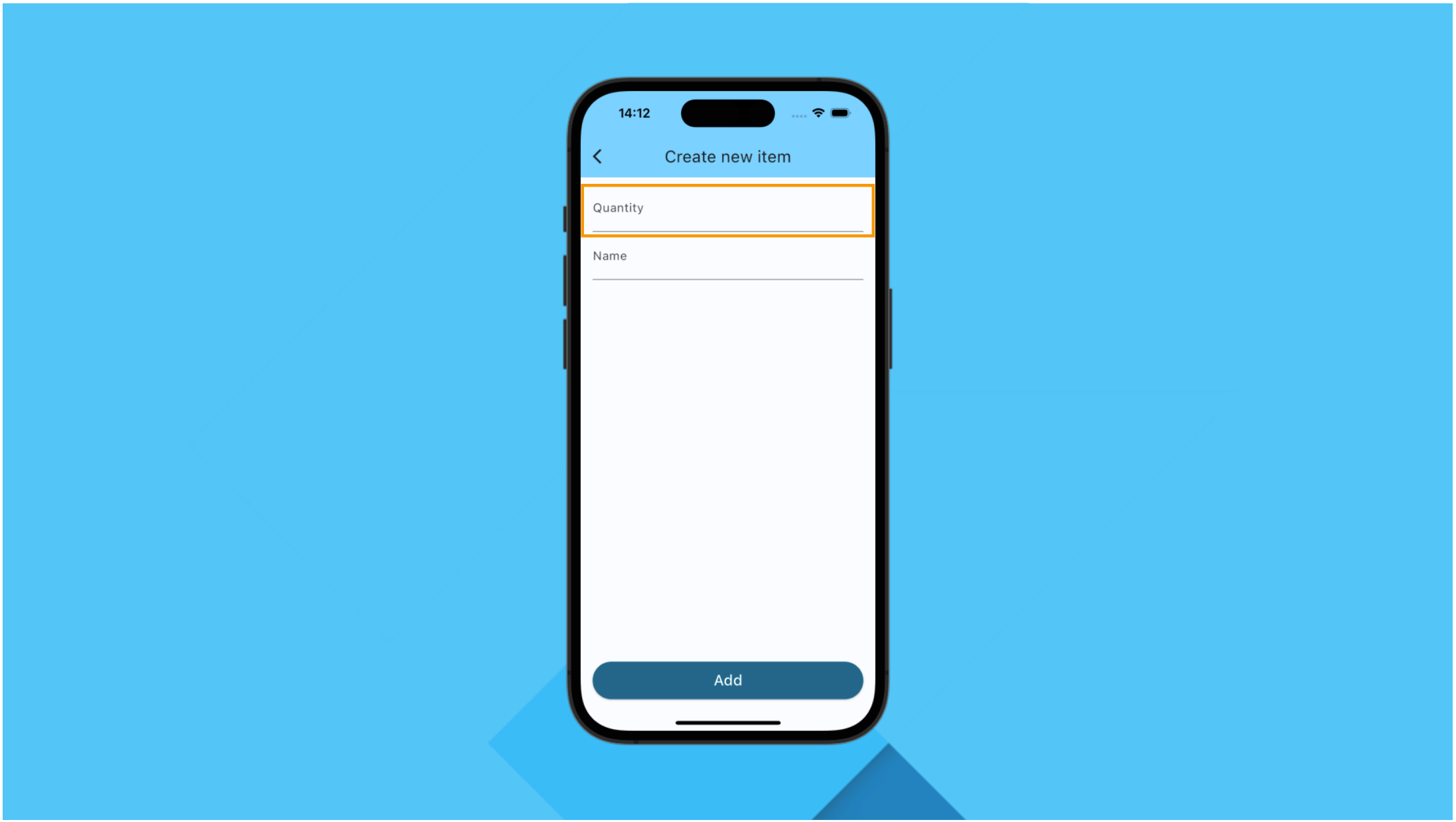


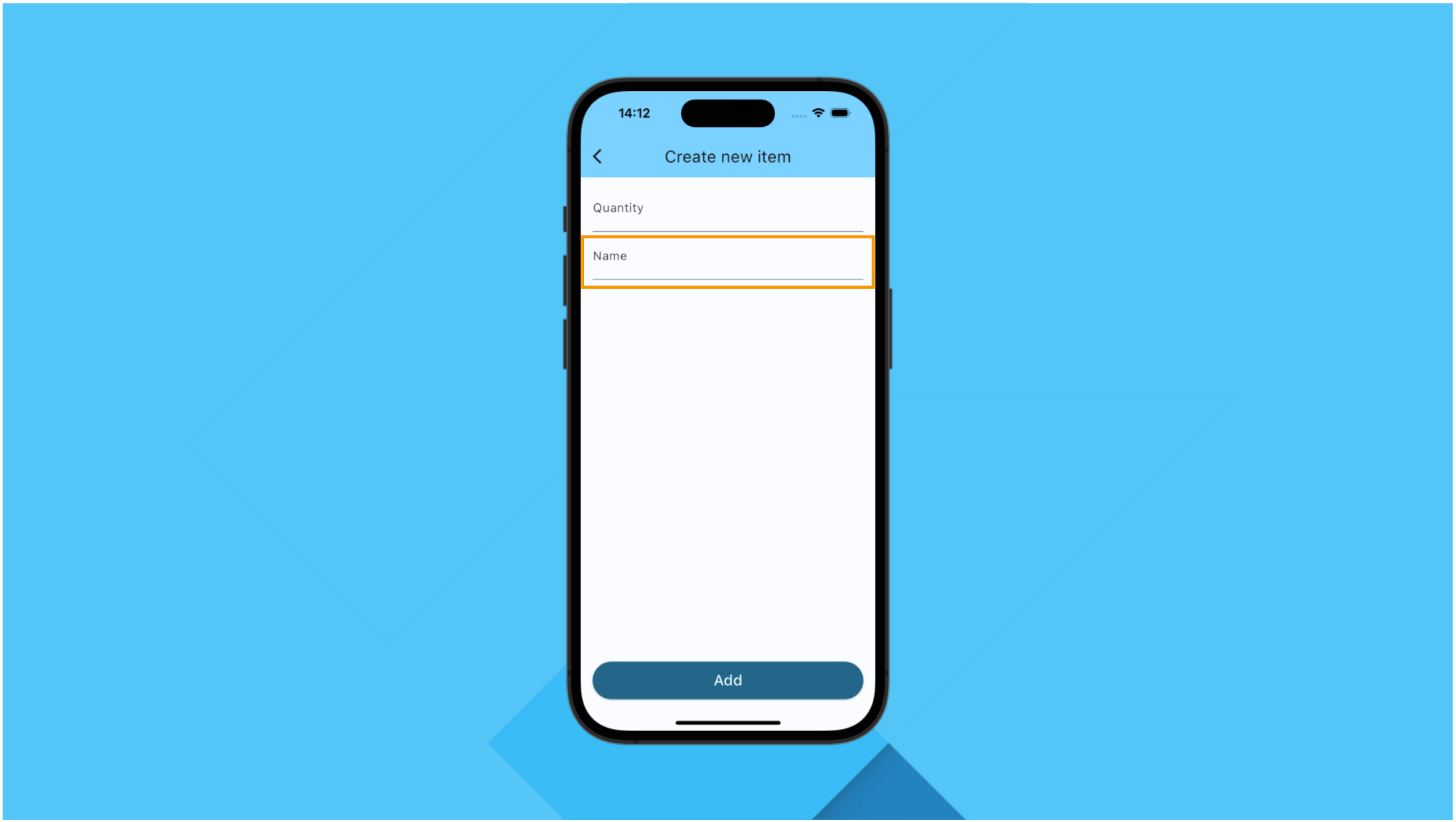
Breaking down the design

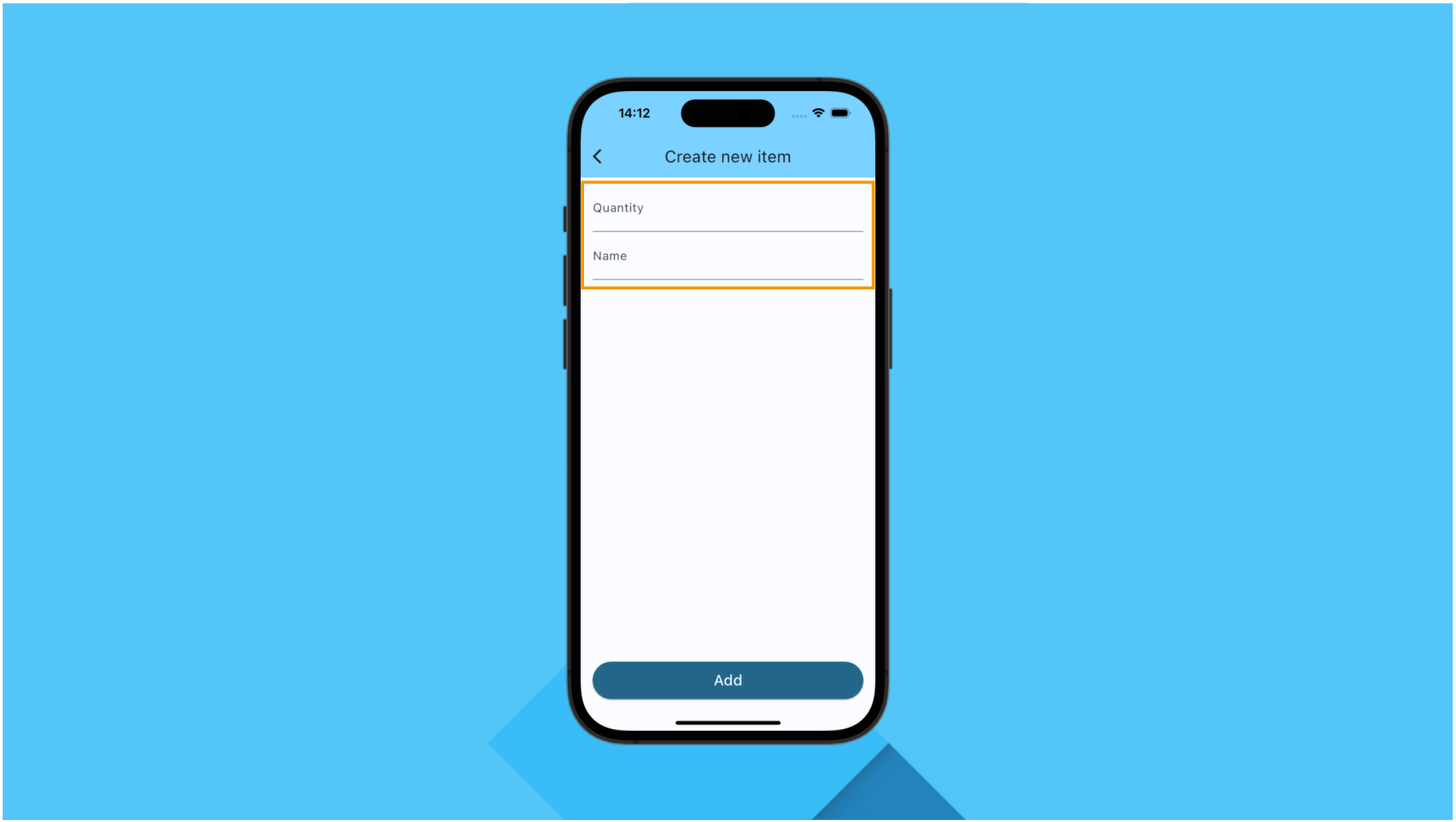
Shopping List App

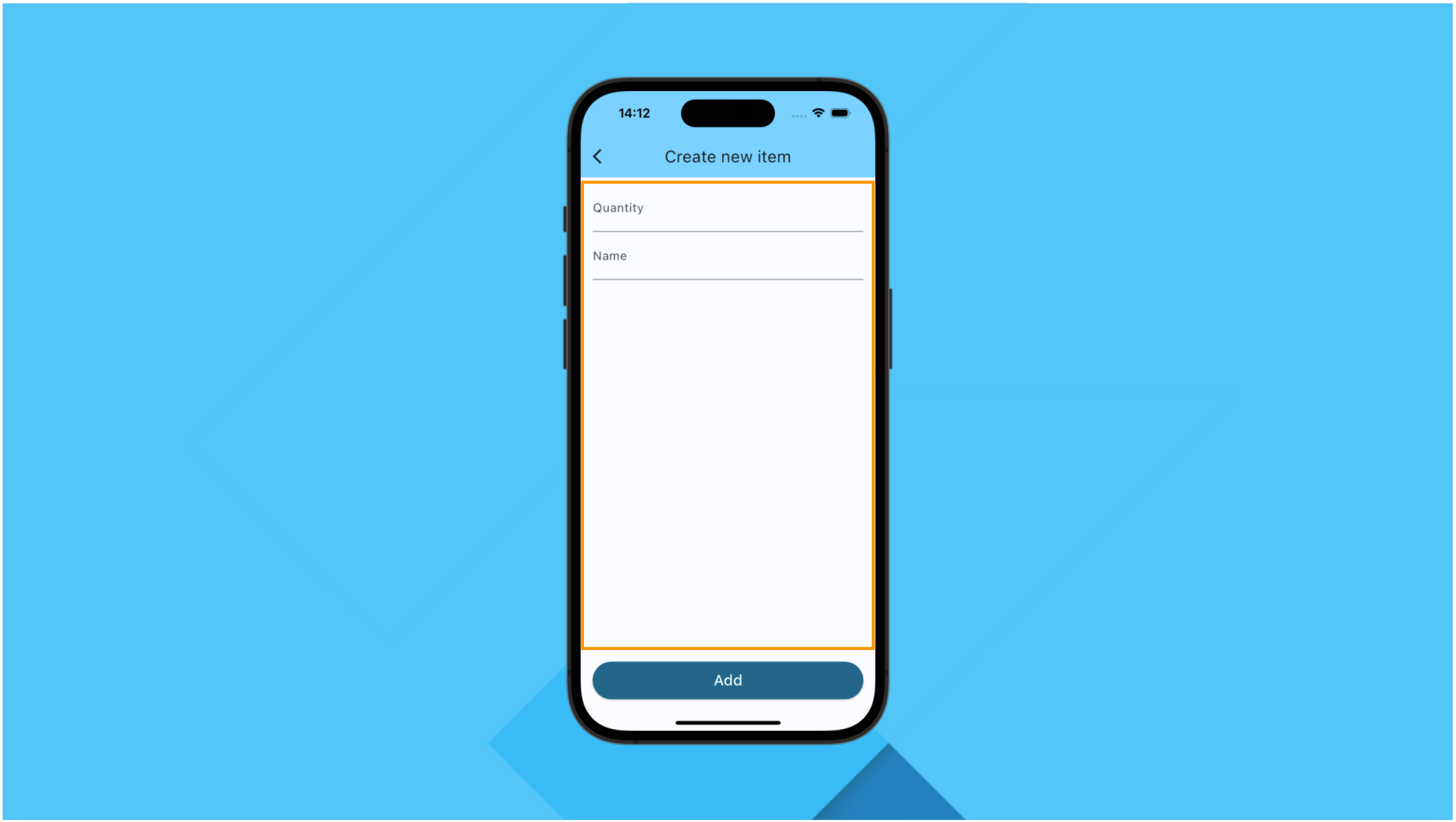

14:12
Create new item
Quantity
Name
Add

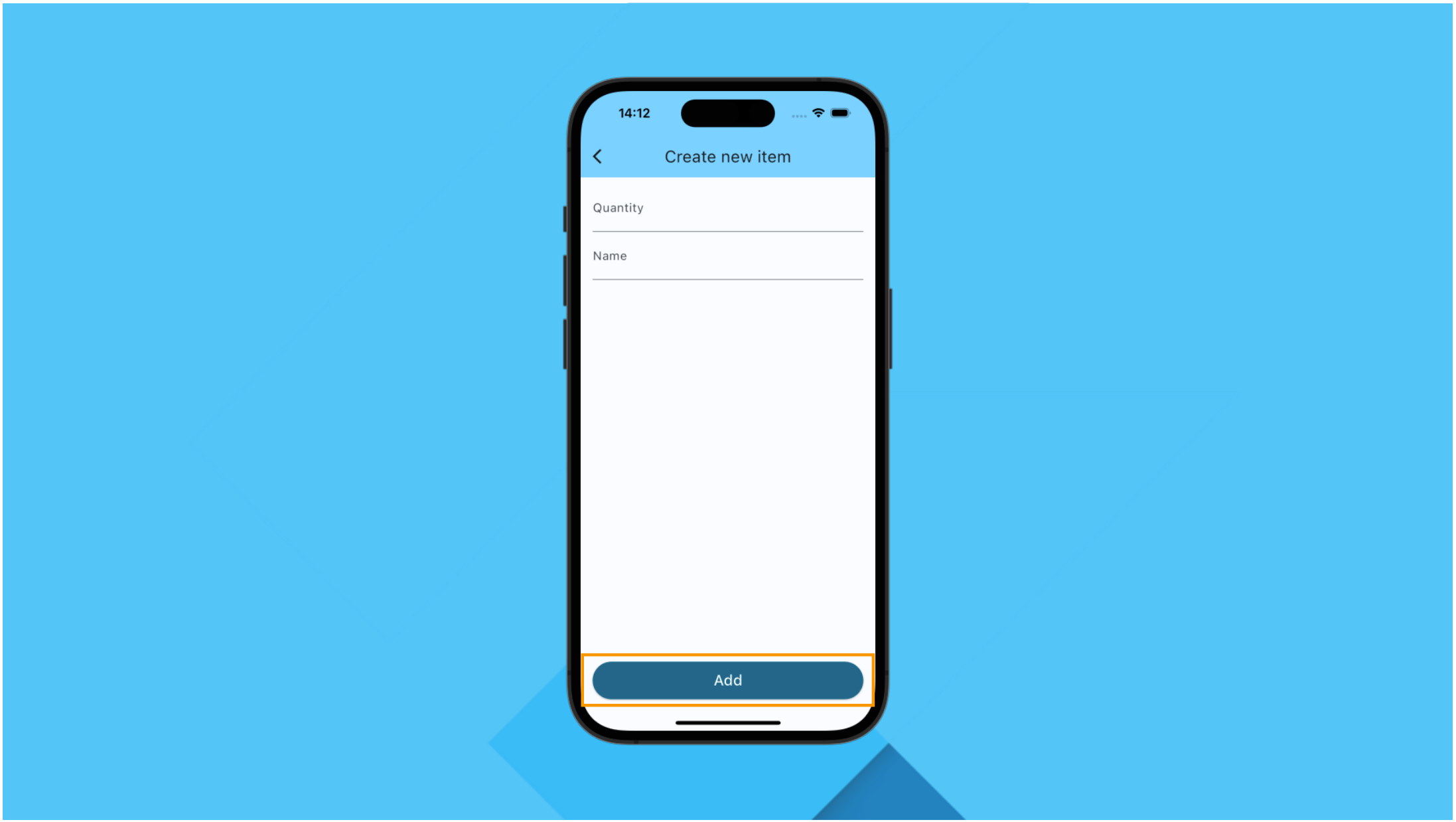

14:12
Create new item
Quantity
Name
Add


14:12
Create new item
Quantity
Name
Add


14:12
Create new item
Quantity
Name
Add


14:12
Create new item
Quantity
Name
Add


14:12
Create new item
Quantity
Name
Add


14:12
Create new item
Quantity
Name
Add

Implementing the page

Shopping List App


```dart
class CreateNewShoppingListItemPage extends
StatelessWidget {
  const CreateNewShoppingListItemPage({
    super.key, required this.shoppingList,
  });
  final ShoppingList shoppingList;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor:
Theme.of(context).colorScheme.inversePrimary,
        title: const Text('Create new item'),
      ),
      body: _CreateNewItemForm(
        shoppingList: shoppingList,
      ),
    );
  }
}
```


```dart
class _CreateNewItemForm extends StatefulWidget {
  const _CreateNewItemForm({
    super.key,
    required this.shoppingList,
  });

  final ShoppingList shoppingList;

  @override
  State<_CreateNewItemForm> createState() =>
_CreateNewItemFormState();
}


class _CreateNewItemFormState extends
State<_CreateNewItemForm> {
  @override
  Widget build(BuildContext context) {
    return const Placeholder();
  }
}
```


```dart
class _CreateNewItemFormState extends
State<_CreateNewItemForm> {
  @override
  Widget build(BuildContext context) {
    return Form(
      child: Column(
        children: [],
      ),
    );
  }
}
```

Column

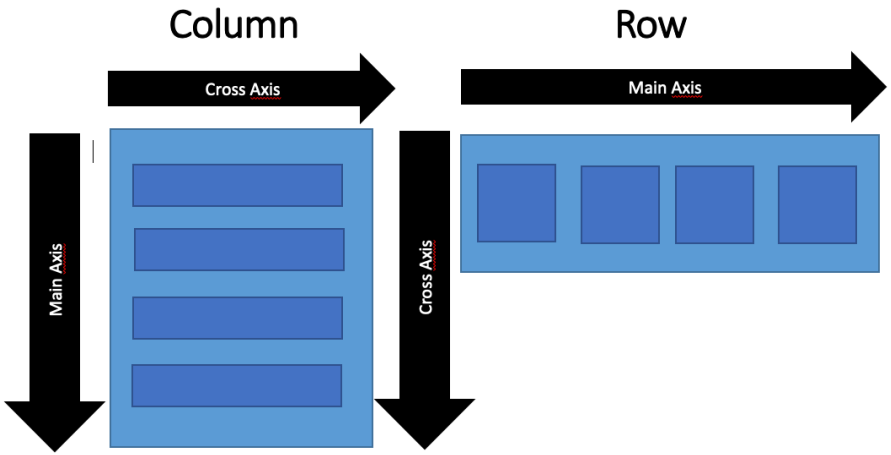
“A widget that displays its children in a vertical array.“

Row

"A widget that displays its children in a horizontal array."

Column/Row

- children: “The widgets below this widget in the tree.“
- mainAxisAlignment: “How the children should be placed along the main axis.“
- mainAxisSize: “How much space should be occupied in the main axis.“
- crossAxisAlignment: “How the children should be placed along the cross axis.“
- see all properties here



Column/Row vs. Flexbox

mainAxisAlignment

- start
- end
- center
- spaceBetween
- spaceAround
- spaceEvenly

justify-content

- start
- end
- center
- space-between
- space-around
- space-evenly

Column/Row vs. Flexbox

crossAxisAlignment

- start
- end
- center
- stretch
- baseline

align-items

- start
- end
- center
- stretch
- baseline

Column/Row vs. Flexbox

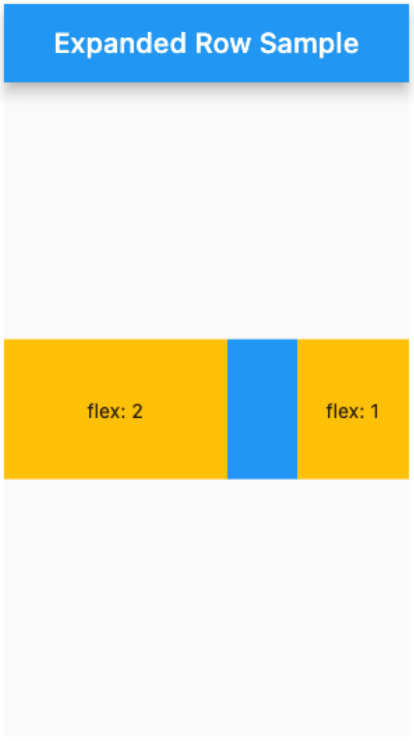
Expanded

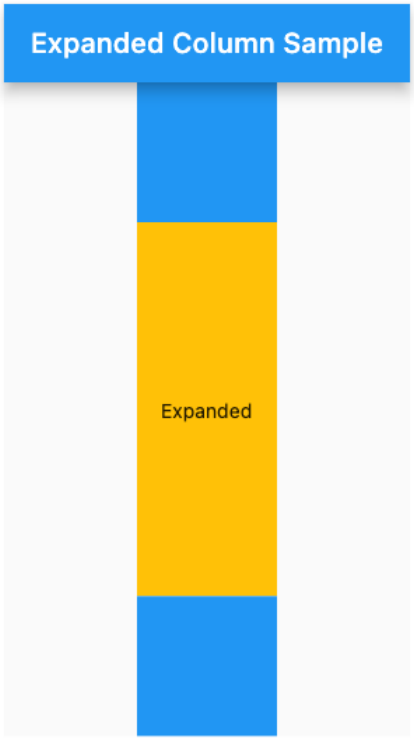
- “A widget that expands a child of Row, Column, or Flex so that the child fills the available space.“

flex

- ”flex CSS shorthand property sets how a flex item will grow or shrink to fit the space available in its flex container.“

Expanded


Expanded Row Sample
flex: 2
flex: 1


Expanded Column Sample
Expanded

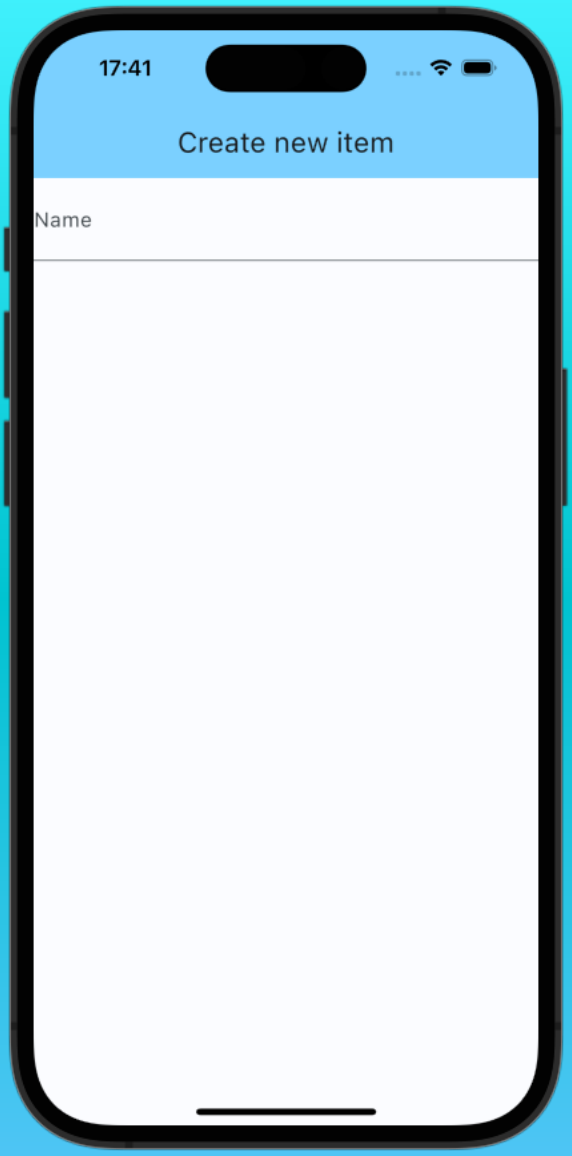

```dart
class _CreateNewItemFormState extends
State<_CreateNewItemForm> {
  @override
  Widget build(BuildContext context) {
    return Form(
      child: Column(
        children: [
          TextFormField(),
        ],
      ),
    );
  }
}
```

TextFormField

“A FormField that contains a TextField. This is a convenience widget that wraps a TextField widget in a FormField.“

TextFormField

- decoration: “The decoration to show around the text field.“
- controller: “Controls the text being edited.“
- validator: “An optional method that validates the input. Returns an error string to display if the input is invalid, or null otherwise.“
- see all properties here




```dart
class _CreateNewItemFormState extends
State<_CreateNewItemForm> {
  @override
  Widget build(BuildContext context) {
    return Form(
      child: Column(
        children: [
          TextFormField(
            decoration: InputDecoration(
              labelText: 'Name',
            ), 👆 give TextField label
          ),
        ],
      ),
    );
  }
}
```

Padding

"A widget that insets its child by the given padding."

Padding

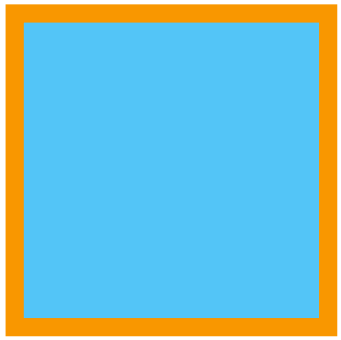
- child: “The widget below this widget in the tree.“
- padding: “The amount of space by which to inset the child.“
- see all properties here

Understanding EdgeInsets

Shopping List App

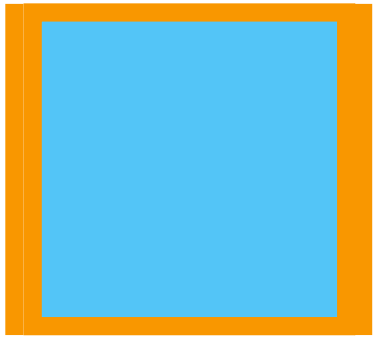
EdgeInsets.all

```
const EdgeInsets.all(8.0);
```



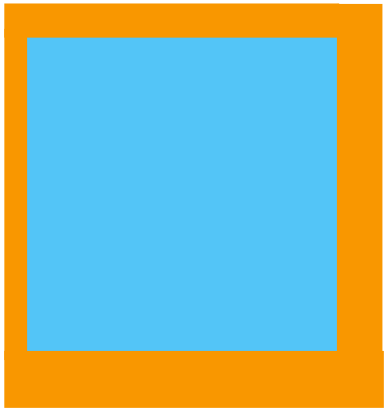
EdgeInsets.symmetric

```
const EdgeInsets.symmetric(
  horizontal: 16,
  vertical: 8,
);
```



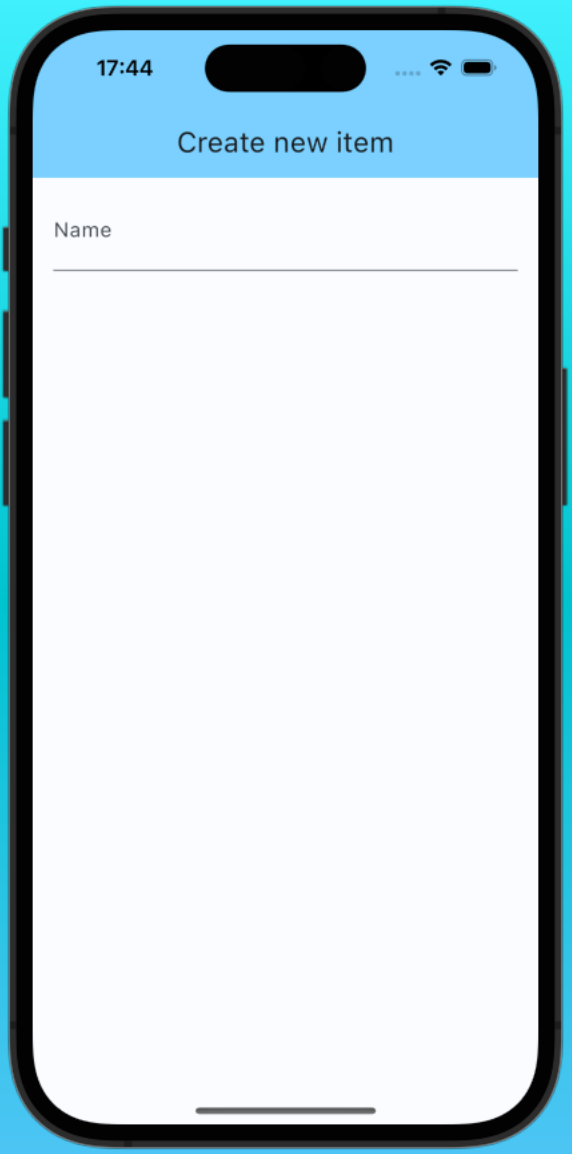
EdgeInsets.fromLTRB

```
const EdgeInsets.fromLTRB(
   10,
   15,
   20,
   25,
);
```

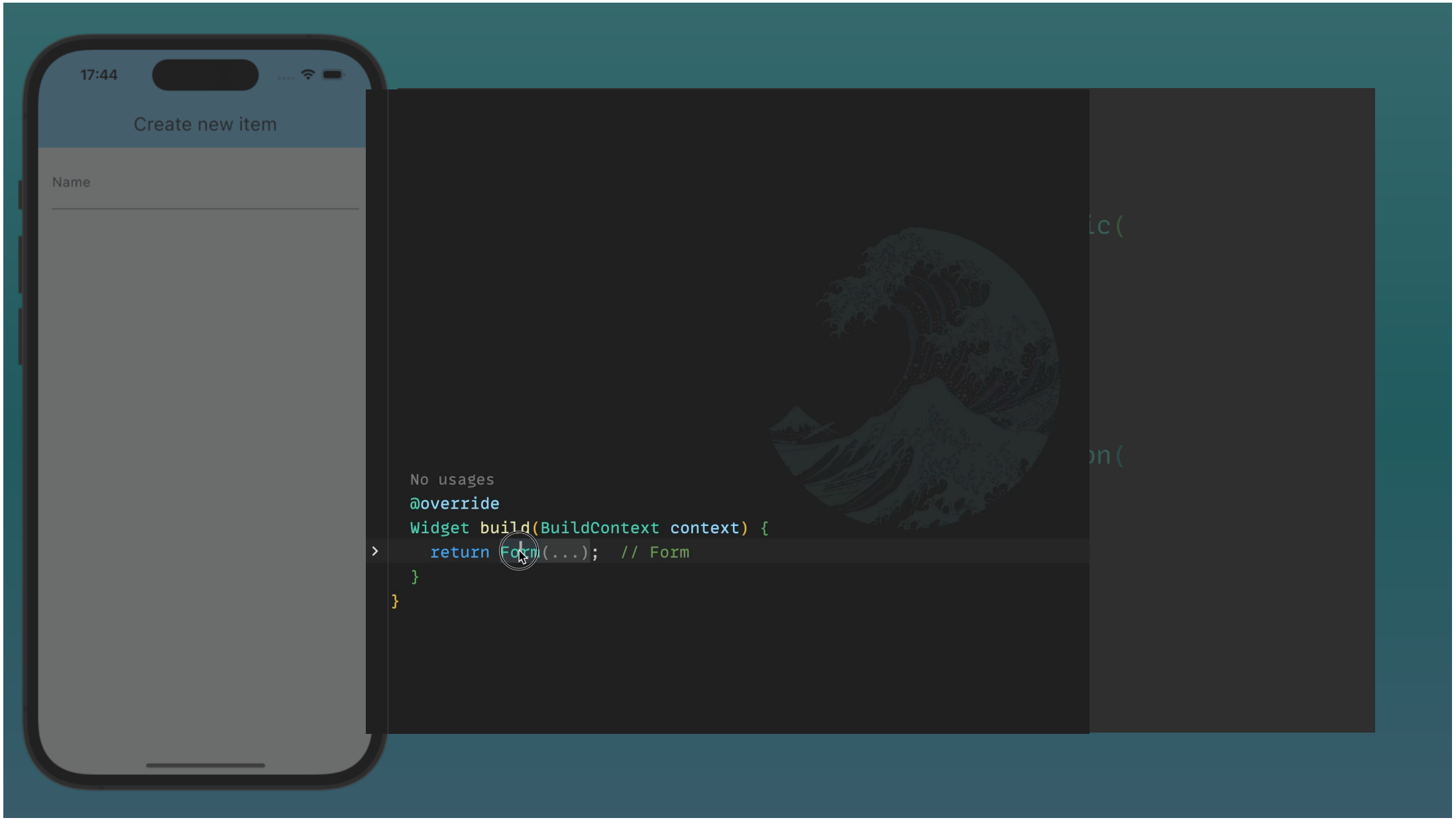


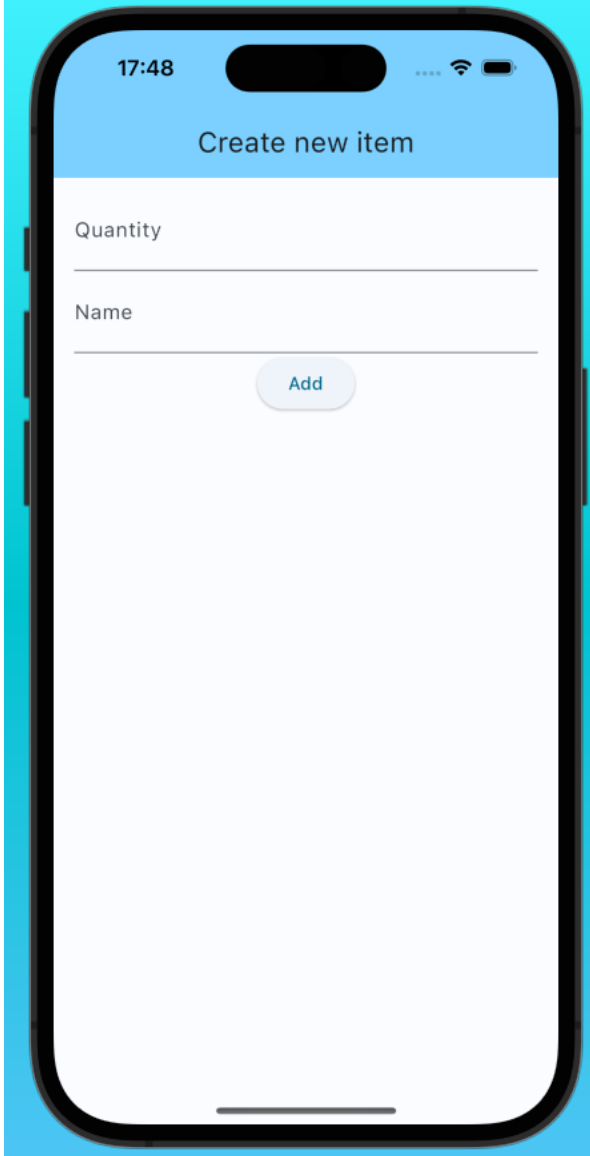
EdgeInsets.only

```
const EdgeInsets.only(left: 10.0);
```

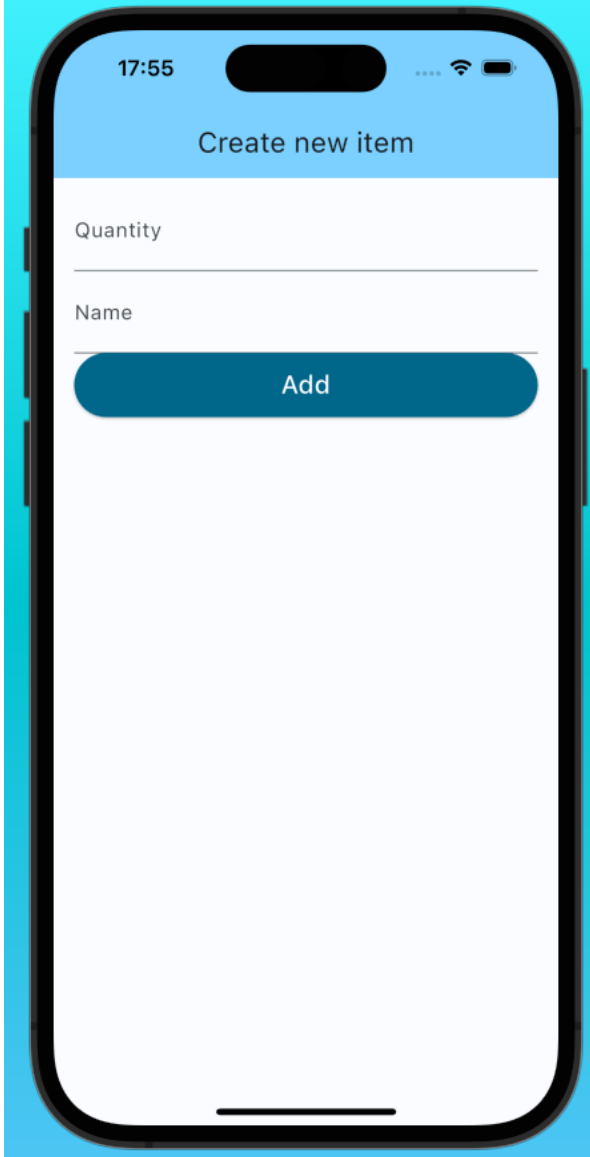



```dart
@override
Widget build(BuildContext context) {
  return Padding(
    padding: const EdgeInsets.symmetric(
      horizontal: 16,
      vertical: 8,
    ),
    child: Form(
      child: Column(
        children: [
          TextFormField(
            decoration: InputDecoration(
              labelText: 'Name',
            ),
          ),
        ],
      ),
    ),
  );
}
```

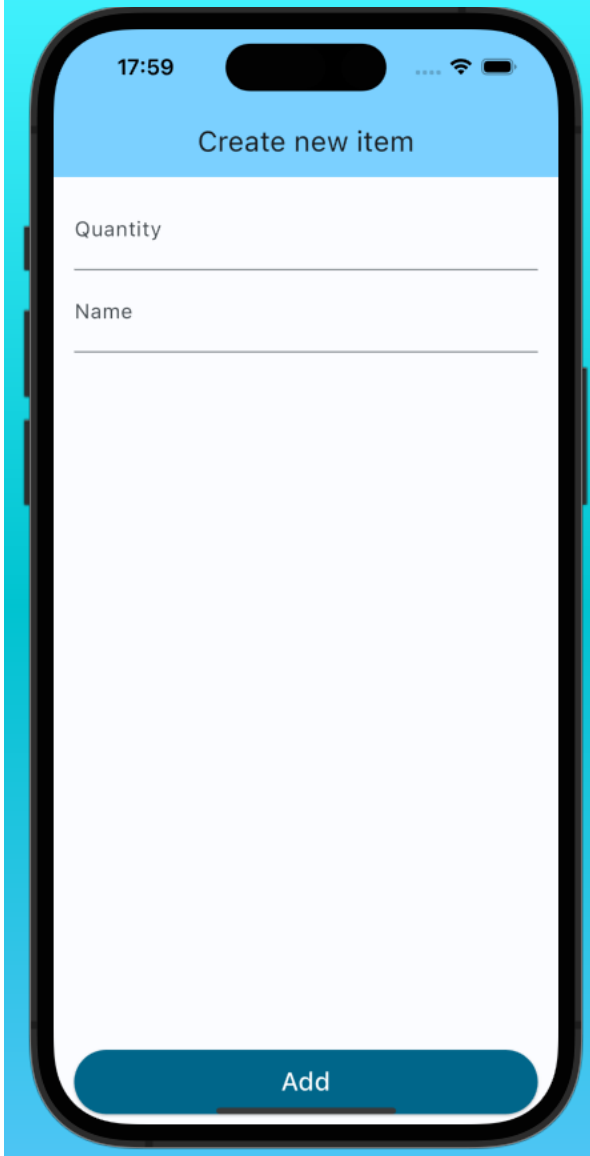

17:44
Create new item
Name
No usages
@override
Widget build(BuildContext context) {
return Form(...); // Form
}
}




```
[
  TextFormField(
    decoration: InputDecoration(
      labelText: 'Quantity',
    ),
  ),
  TextFormField(
    decoration: InputDecoration(
      labelText: 'Name',
    ),
  ),
  ElevatedButton(
    onPressed: () {},
    child: const Text('Add'),
  ),
]
```



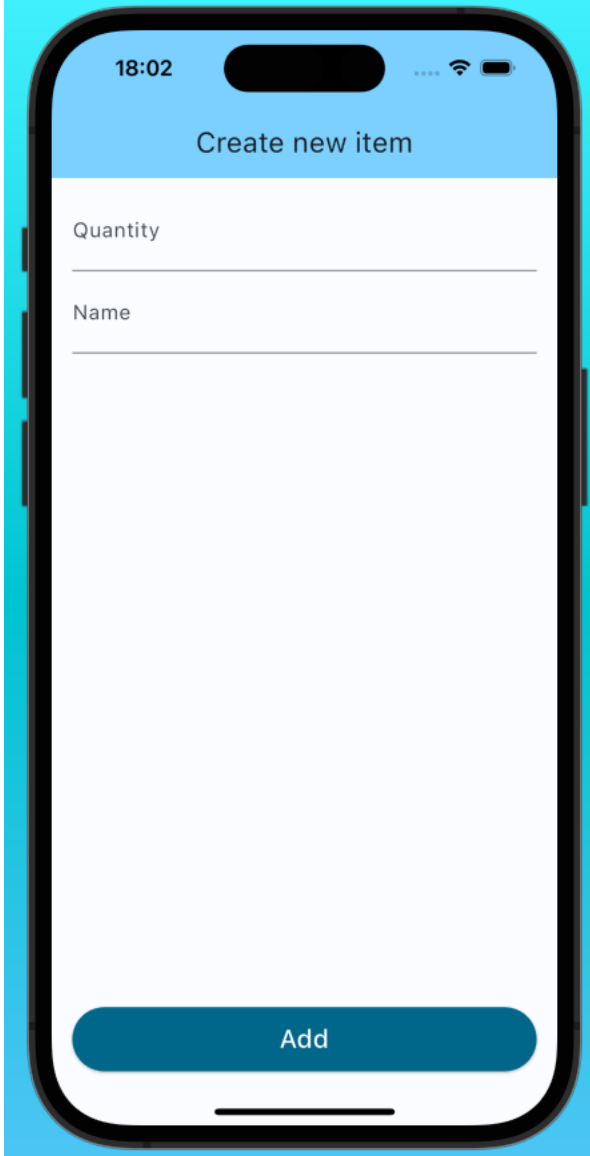

```dart
ElevatedButton(
  style: ElevatedButton.styleFrom(
    backgroundColor:
Theme.of(context).colorScheme.primary,
    foregroundColor:
Theme.of(context).colorScheme.onPrimary,
    minimumSize: const Size.fromHeight(50.0),
  ),
  onPressed: () {},
  child: const Text(
    'Add',
    style: TextStyle(fontSize: 20),
  ),
),
```




```
[
  Expanded(
    child: Column(
      children: [
        TextFormField(
          ...
        ),
        TextFormField(
          ...
        ),
      ],
    ),
  ),
  ElevatedButton(
    ...
  ),
]
```

SafeArea

“A widget that insets its child by sufficient padding to avoid intrusions by the operating system.“




```dart
return SafeArea(
  child: ...
);
```

Capturing Input

Shopping List App

Capturing Input

- Two viable approaches

1. Using a state variable and setting it in the onChanged callback of the TextFormField
2. Using a TextEditingController

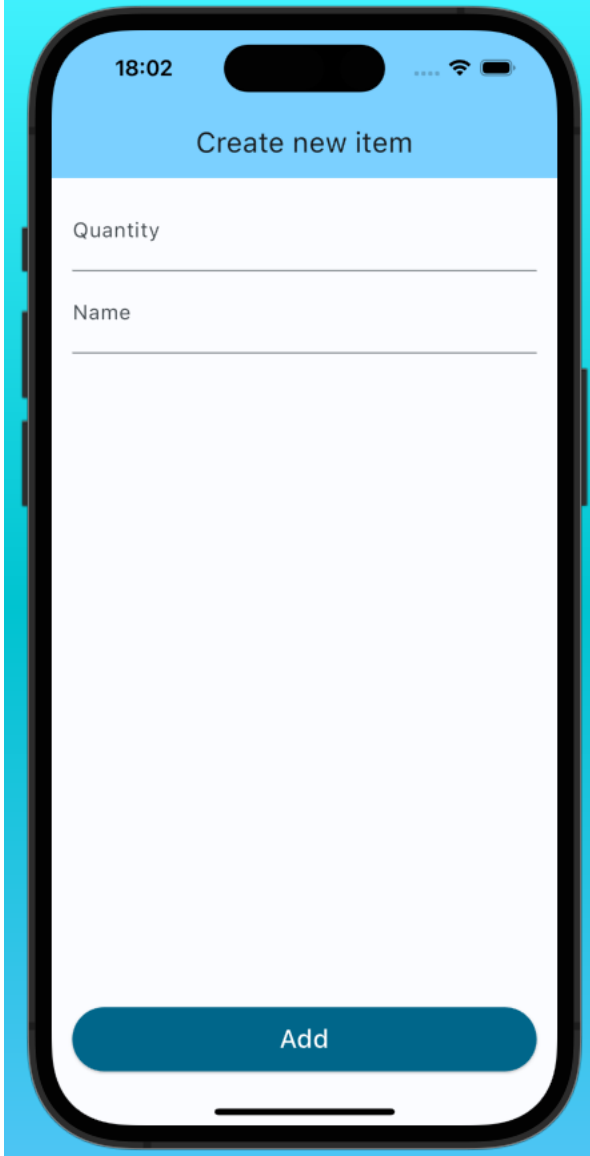
Capturing Input

State variables

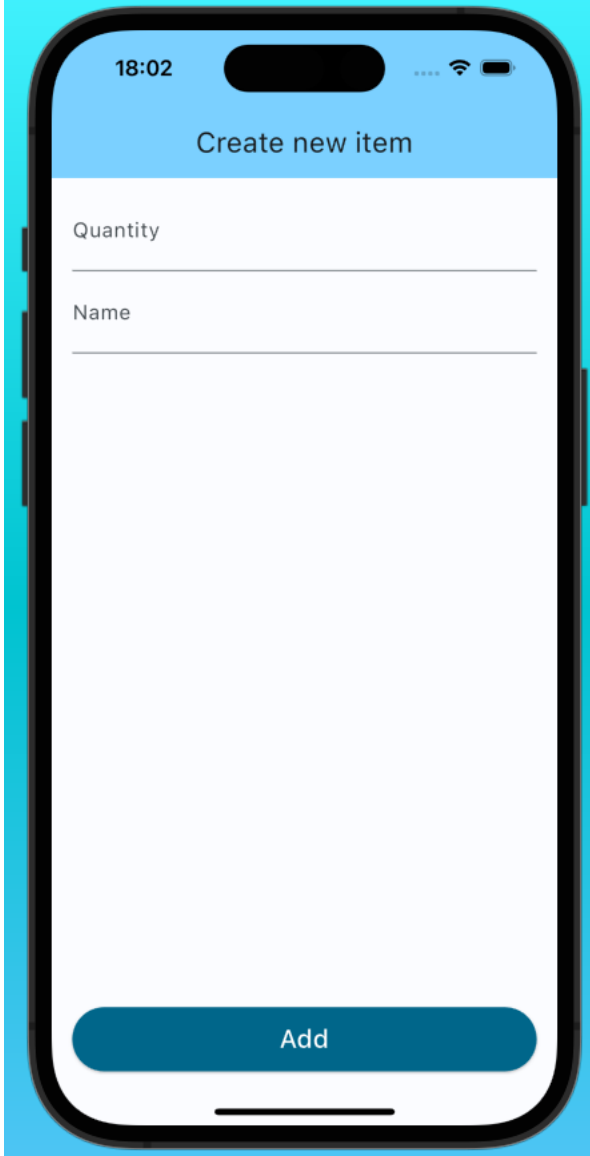
- Rerenders the whole widget tree when the input changes
- Useful when the input is displayed somewhere else than the TextField itself

TextEditingController

- Only rerenders the TextField
- Requires disposing and initializing




```dart
class _CreateNewItemFormState extends
State<_CreateNewItemForm> {
  late final TextEditingController
_quantityController;
  late final TextEditingController
_nameController;
  // tells the compiler that _nameController is initialized before
} being used
```




```dart
class _CreateNewItemFormState extends
State<_CreateNewItemForm> {
  late final TextEditingController
_quantityController;
  late final TextEditingController
_nameController;

  @override
  void initState() {
    super.initState();
    _quantityController =
TextEditingController();
    _nameController = TextEditingController();
  }

  ...

}
```

Lifecycle methods of a StatefulWidget

Shopping List App

Lifecycle methods of a StatefulWidget (processed in order)

1. createState()
2. mounted = true
3. initState()
4. didChangeDependencies()
5. build()
6. didUpdateWidget()
7. setState()
8. deactivate()
9. dispose()
10. mounted = false

initState()

- Is the first method called after the creation of the widget
- Only called once!
- Must call super.initState() before doing anything else
- Useful for:
  - Initializing data that relies on the BuildContext
  - Initializing properties that rely on properties from the widget itself
  - Subscribing to any object that could change the data on the widget

build()

- Called on every render => Called very often!
- Must return a widget

dispose()

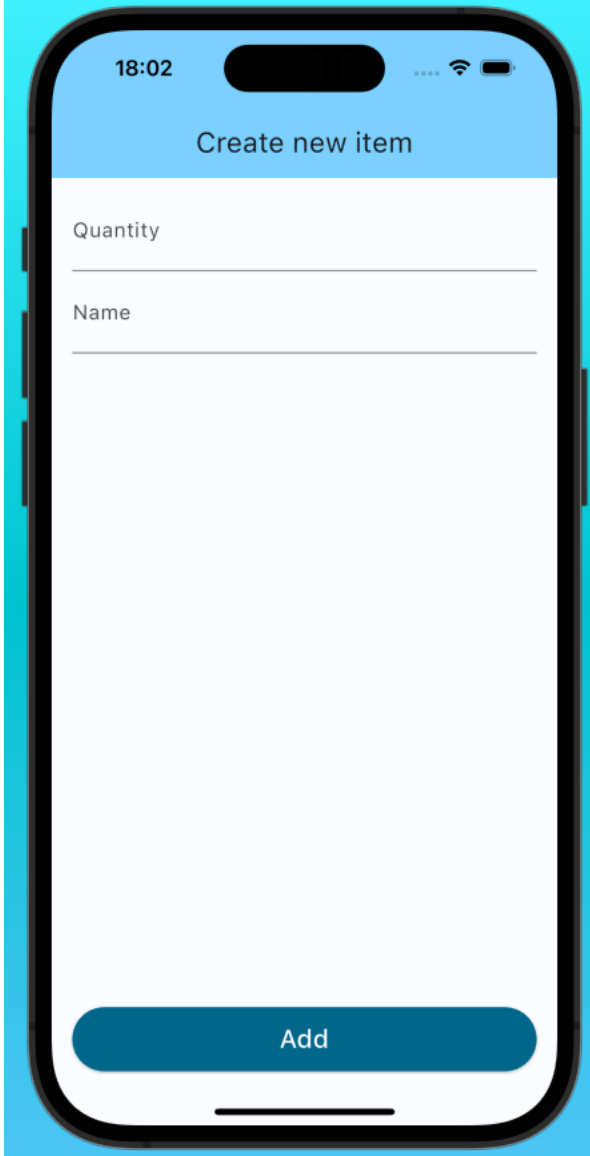
- Called when the state is permantently removed from the tree
- Needs to call super.dispose() after everything else
- Useful for:
  - Unsubscribing
  - Cancelling animations
  - Calling dispose methods of state members (e.g. TextEditingController)

Lifecycle methods of a StatefulWidget

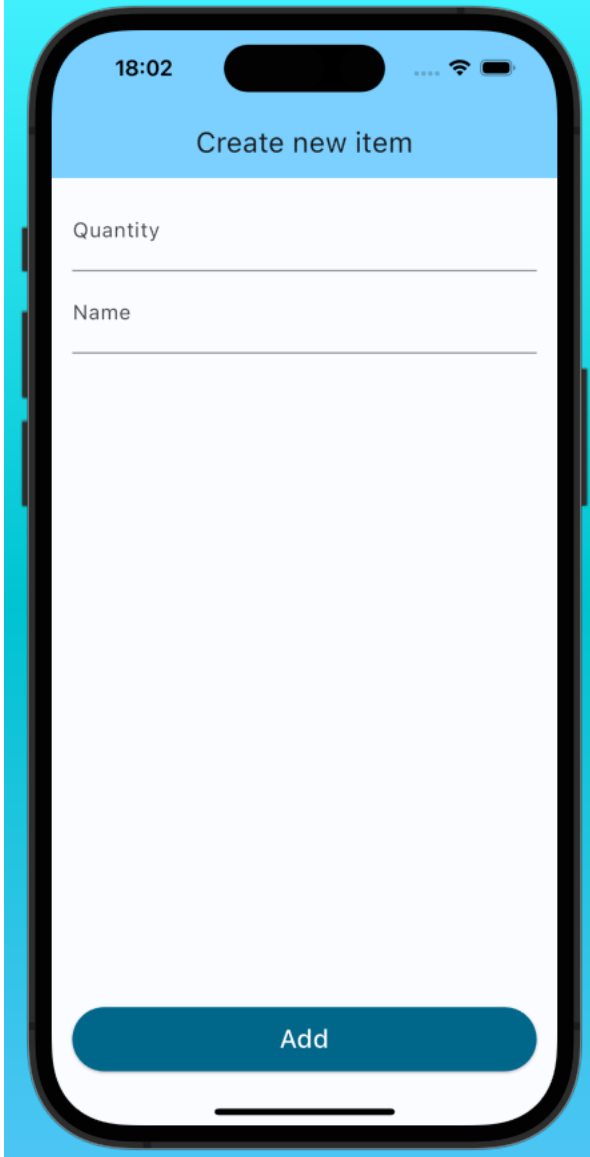
1. createState()
2. mounted = true
3. initState()
4. didChangeDependencies()
5. build()
6. didUpdateWidget()
7. setState()
8. deactivate()
9. dispose()
10. mounted = false

Handling the lifecycle of a TextEditingController

Shopping List App



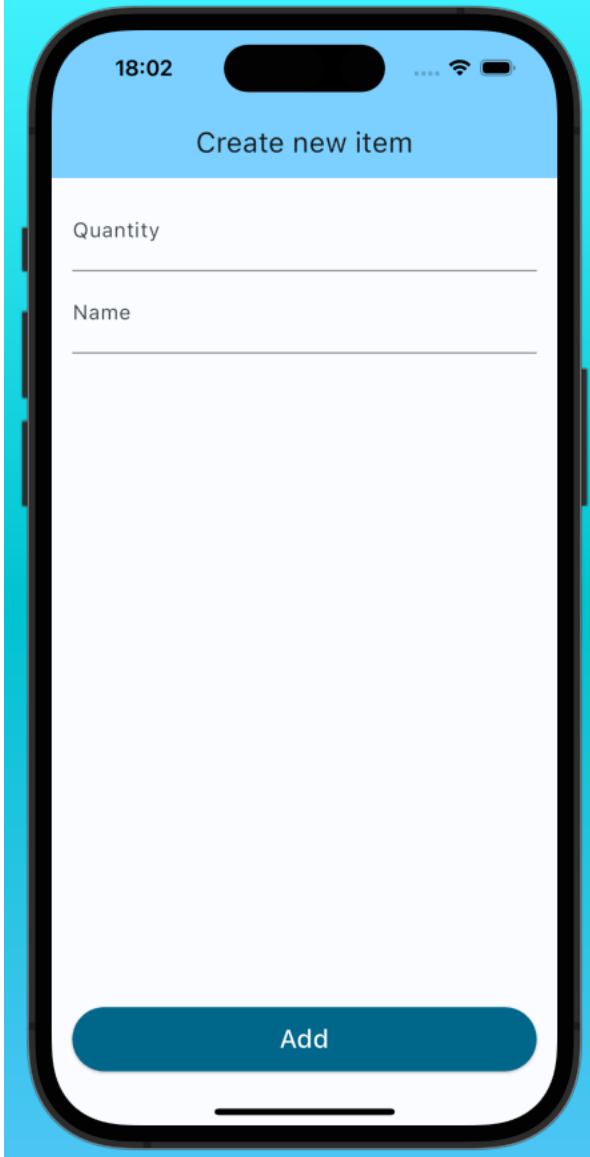

```dart
class _CreateNewItemFormState extends
State<_CreateNewItemForm> {
  late final TextEditingController
_quantityController;
  late final TextEditingController _nameController;

  @override
  void initState() {
    super.initState();
    _quantityController = TextEditingController();
    _nameController = TextEditingController();
  }

  @override
  void dispose() {
    _quantityController.dispose();
    _nameController.dispose();
    super.dispose();
  }
  ...
}
```



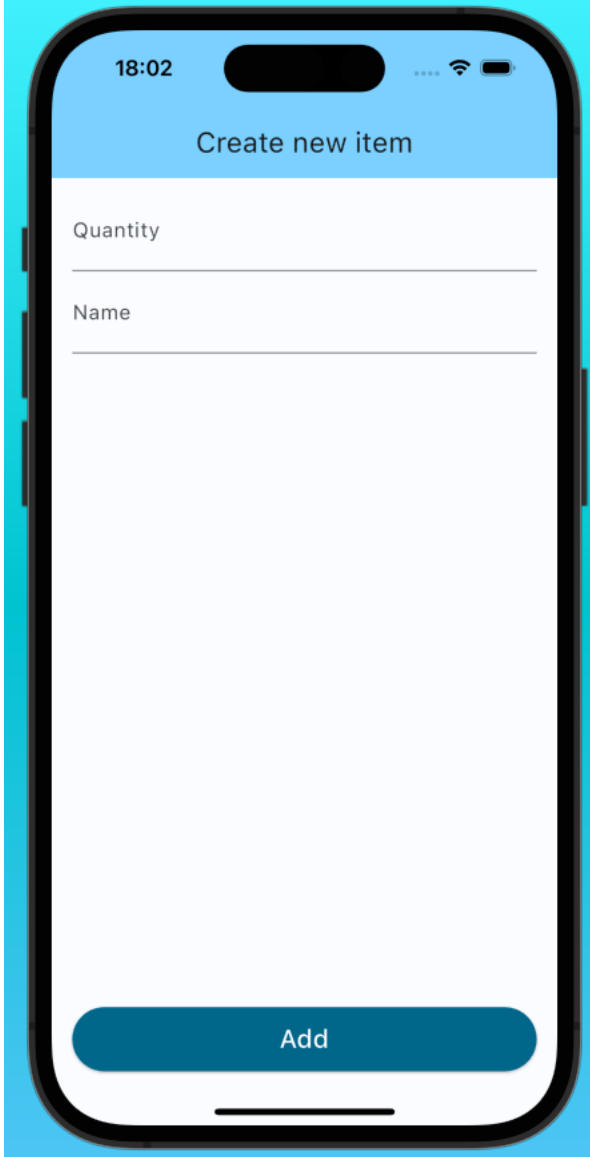

```
[
  TextFormField(
    controller: _quantityController,
    decoration: InputDecoration(
      labelText: 'Quantity',
    ),
  ),
  TextFormField(
    controller: _nameController,
    decoration: InputDecoration(
      labelText: 'Name',
    ),
  ),
]
```

Shopping list app

Adding validation

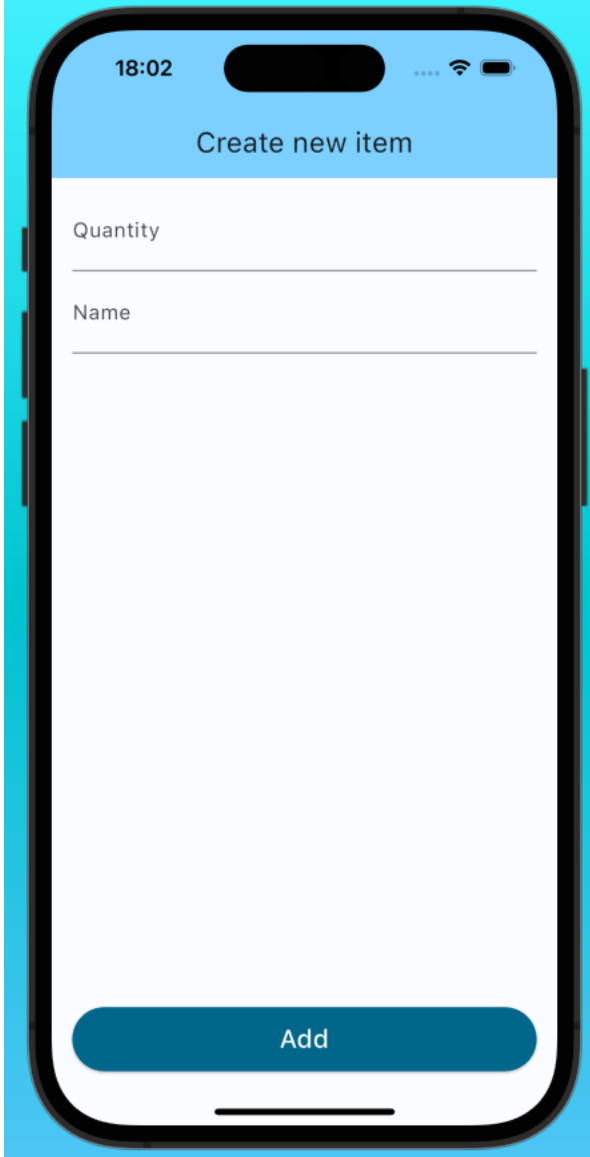



```dart
[
  TextFormField(
    controller: _quantityController,
    keyboardType: TextInputType.numberWithOptions(
      decimal: false,
      signed: false,
    ),
    decoration: InputDecoration(
      labelText: 'Quantity',
    ),
  ),
  TextFormField(
    validator: (value) {
      if (value == null || value.isEmpty) {
        return 'Please enter a name';
      }
      return null;
    },
    controller: _nameController,
    decoration: InputDecoration(
      labelText: 'Name',
    ),
  ),
]
```

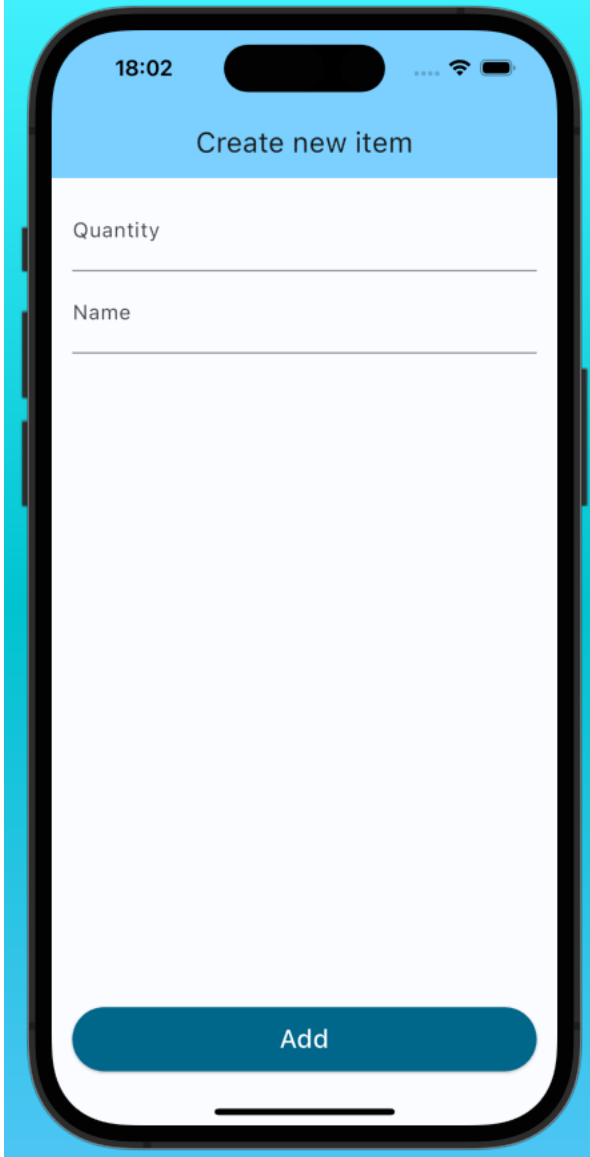



```dart
final _formKey = GlobalKey<FormState>();

@override
Widget build(BuildContext context) {
  return SafeArea(
    child: Padding(
      padding: ...,
      child: Form(
        key: _formKey,
        child: ...
      ),
    ),
  );
}
```

👆uniquely identifies the form and allows validation



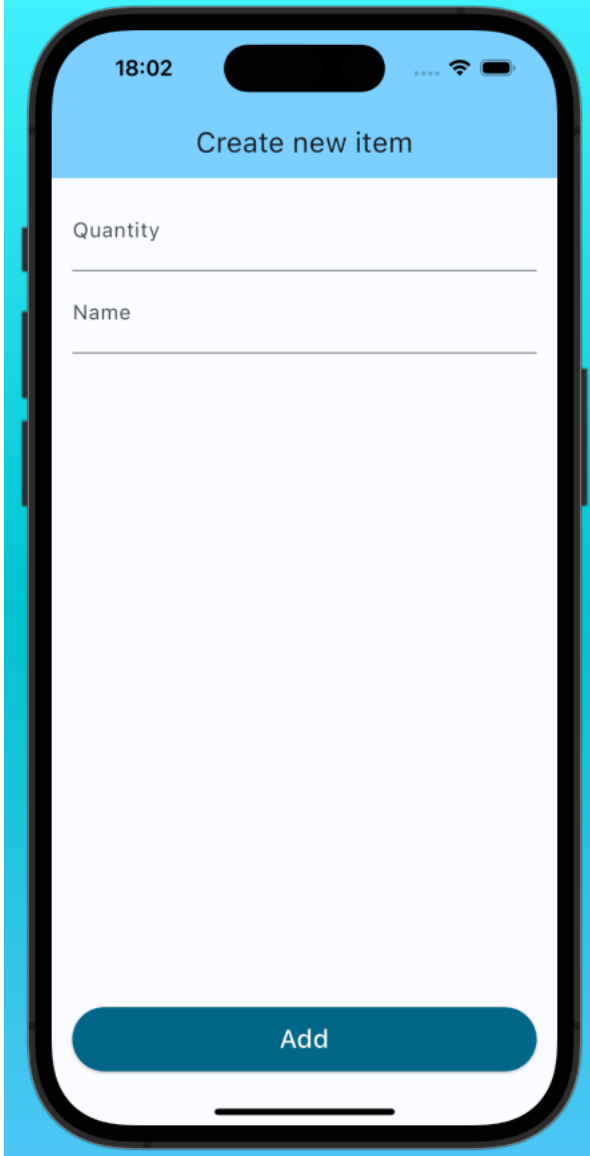

```dart
ElevatedButton(
  style: ElevatedButton.styleFrom(
    backgroundColor:
Theme.of(context).colorScheme.primary,
    foregroundColor:
Theme.of(context).colorScheme.onPrimary,
    minimumSize: const Size.fromHeight(50.0),
  ),
  onPressed: () {},
  child: const Text(
    'Add',
    style: TextStyle(fontSize: 20),
  ),
),
```




```dart
ElevatedButton(
  style: ...,
  onPressed: () {
    if (!_formKey.currentState!.validate()) {
      return;
    }
  },
  child: ...,
),
```

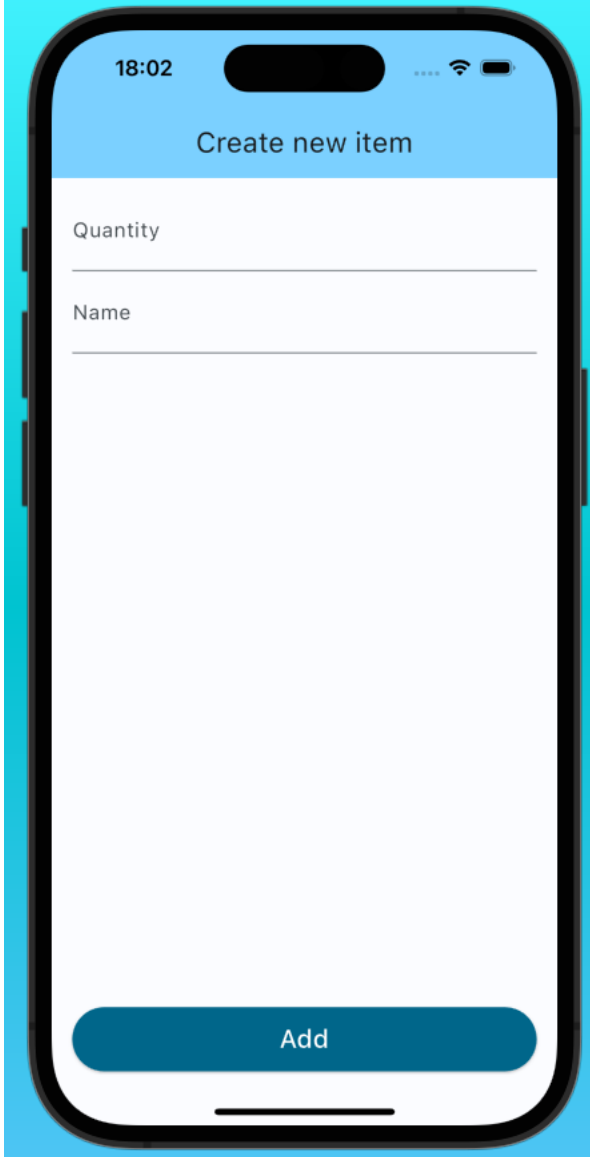
Extracting Input

Shopping List App




```dart
ElevatedButton(
  style: ...,
  onPressed: () {
    if (!_formKey.currentState!.validate()) {
      return;
    }
    final quantity =
int.tryParse(_quantityController.text) ?? 1;
    final name = _nameController.text;
  },
  child: ...,
),
```

Null operator 👇




```dart
ElevatedButton(
  style: ...,
  onPressed: () {
    if (!_formKey.currentState!.validate()) {
      return;
    }
    final quantity =
int.tryParse(_quantityController.text) ?? 1;
    final name = _nameController.text;
    widget.shoppingList.addItem(
      name: name,
      quantity: quantity,
    );
  },
  child: ...,
),
```

Navigation to the new Page

Shopping List App

Navigators

Navigator 1.0

- Imperative API
- Limited control over navigation stack
- Easy to use
- Sufficient for many usecases

Navigator 2.0

- Declarative API
- Full control
- Complex
- Mostly used via third party packages like auto_route or go_router

Navigators

Navigator 1.0

- Imperative API
- Limited control over navigation stack
- Easy to use
- Sufficient for many usecases

👈 We will use the simpler approach

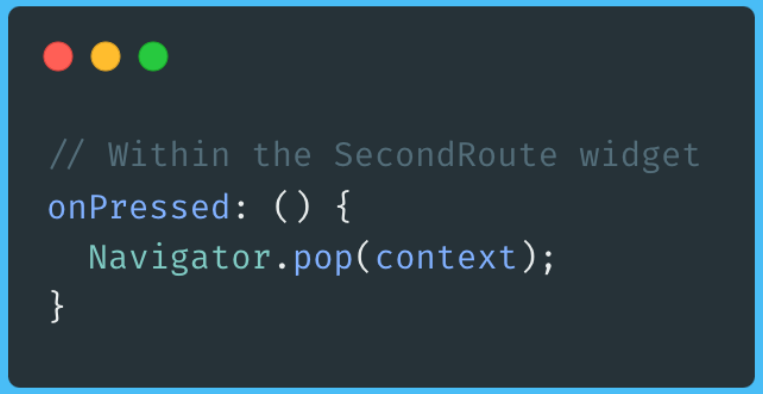
Navigator 2.0

- Declarative API
- Full control
- Complex
- Mostly used via third party packages like auto_route or go_router

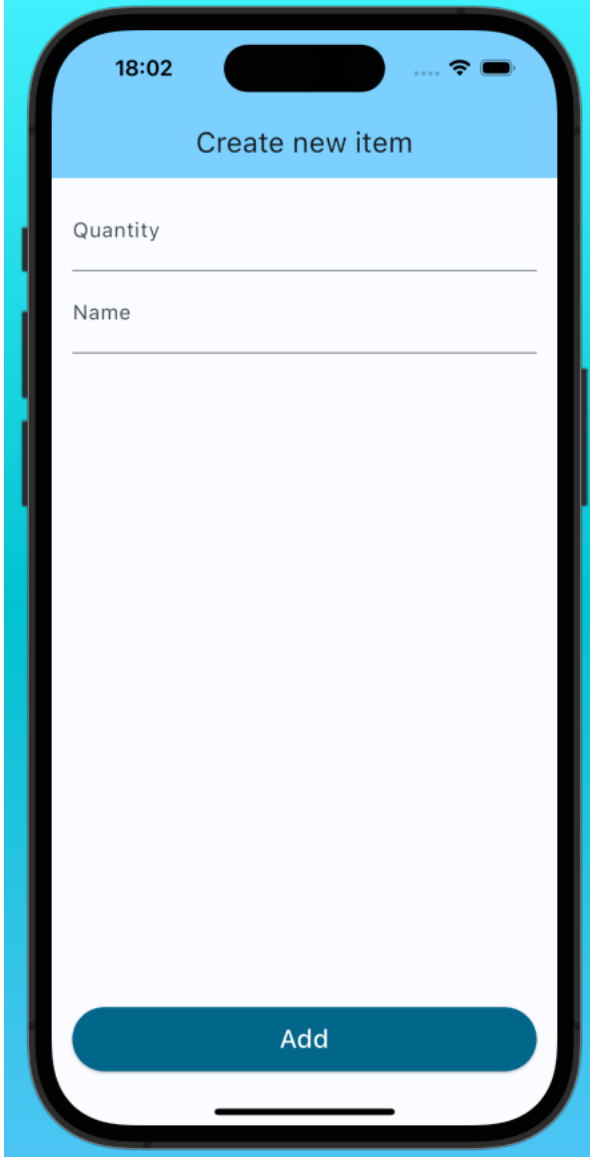
Pushing new pages

```
// Within the `FirstRoute` widget
onPressed: () {
  Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => const SecondRoute()),
  );
}
```

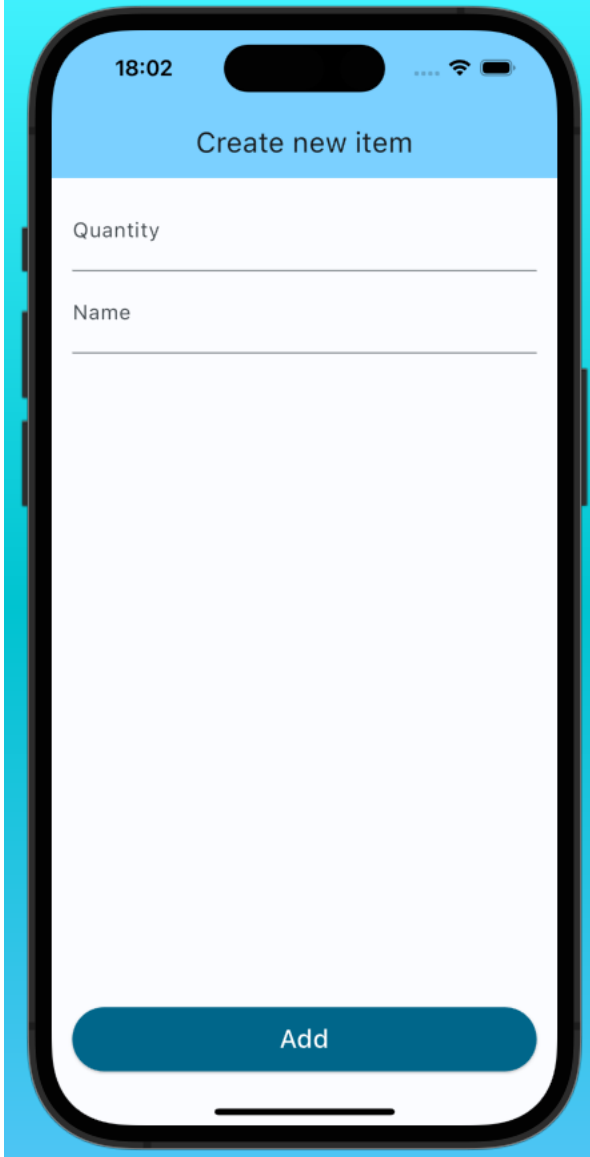
Popping pages



```
// Within the SecondRoute widget
onPressed: () {
  Navigator.pop(context);
}
```

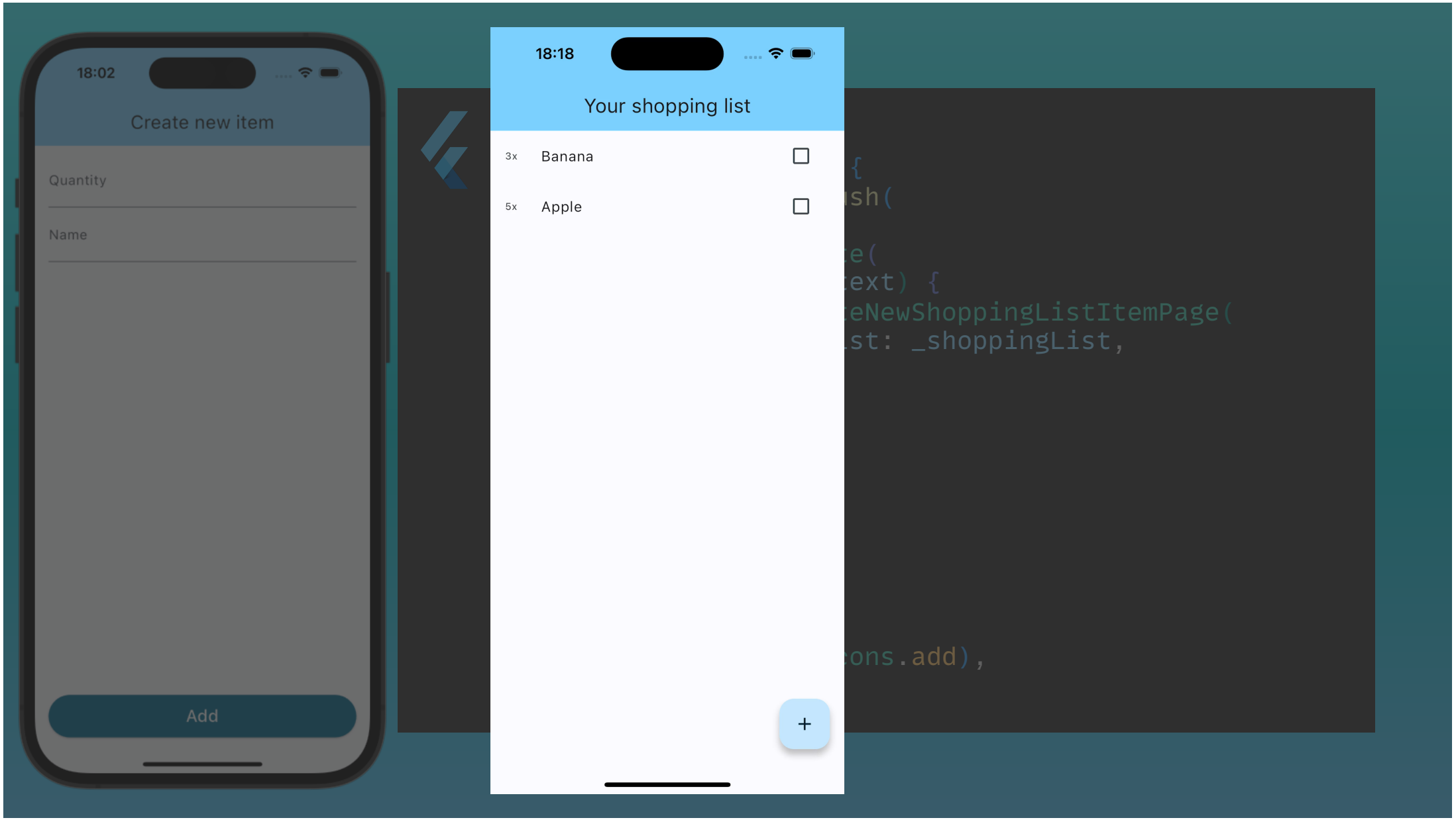



```dart
ElevatedButton(
  style: ...,
  onPressed: () {
    if (!_formKey.currentState!.validate()) {
      return;
    }
    final quantity =
int.tryParse(_quantityController.text) ?? 1;
    final name = _nameController.text;
    widget.shoppingList.addItem(
      name: name,
      quantity: quantity,
    );
    Navigator.pop(context);
  },
  child: ...,
),
```



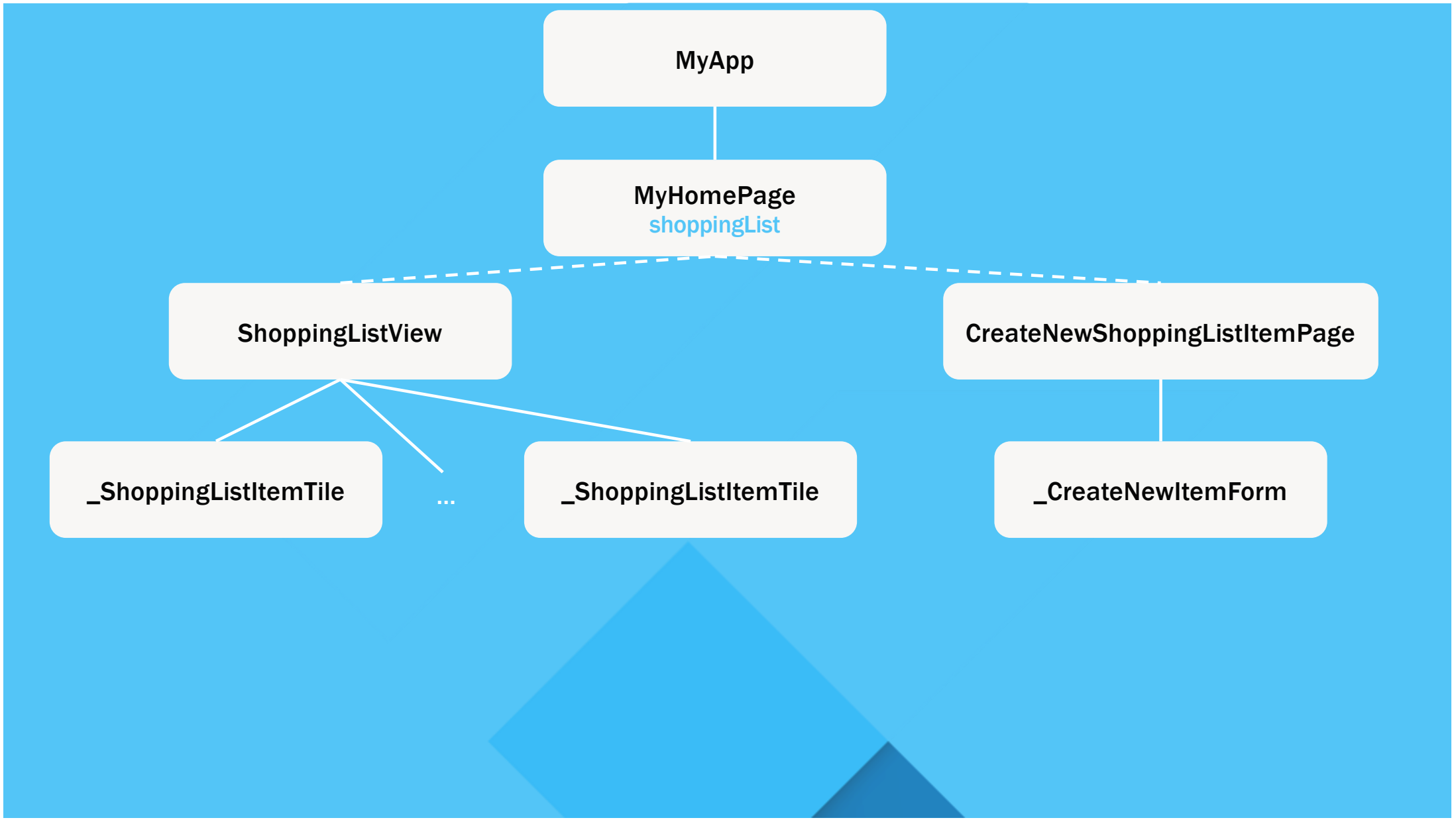

```dart
FloatingActionButton(
  onPressed: () async {
    await Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) {
          return CreateNewShoppingListItemPage(
            shoppingList: _shoppingList,
          );
        },
      ),
    );
    if(mounted) {
      setState(() {
        // rerender
      });
    }
  },
  child: const Icon(Icons.add),
)
```

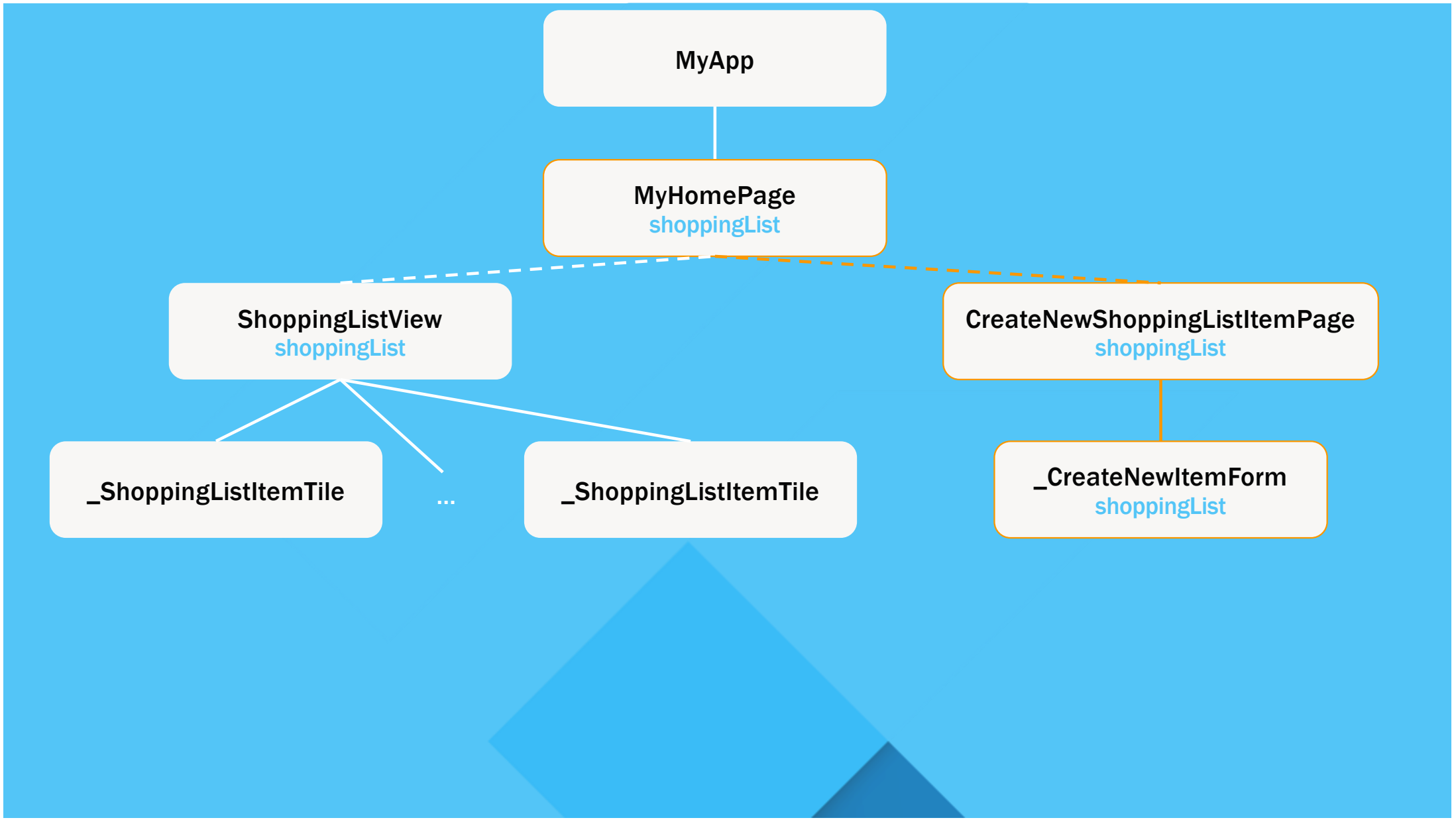
👈 check if the widget is still mounted after an asynchronous operation

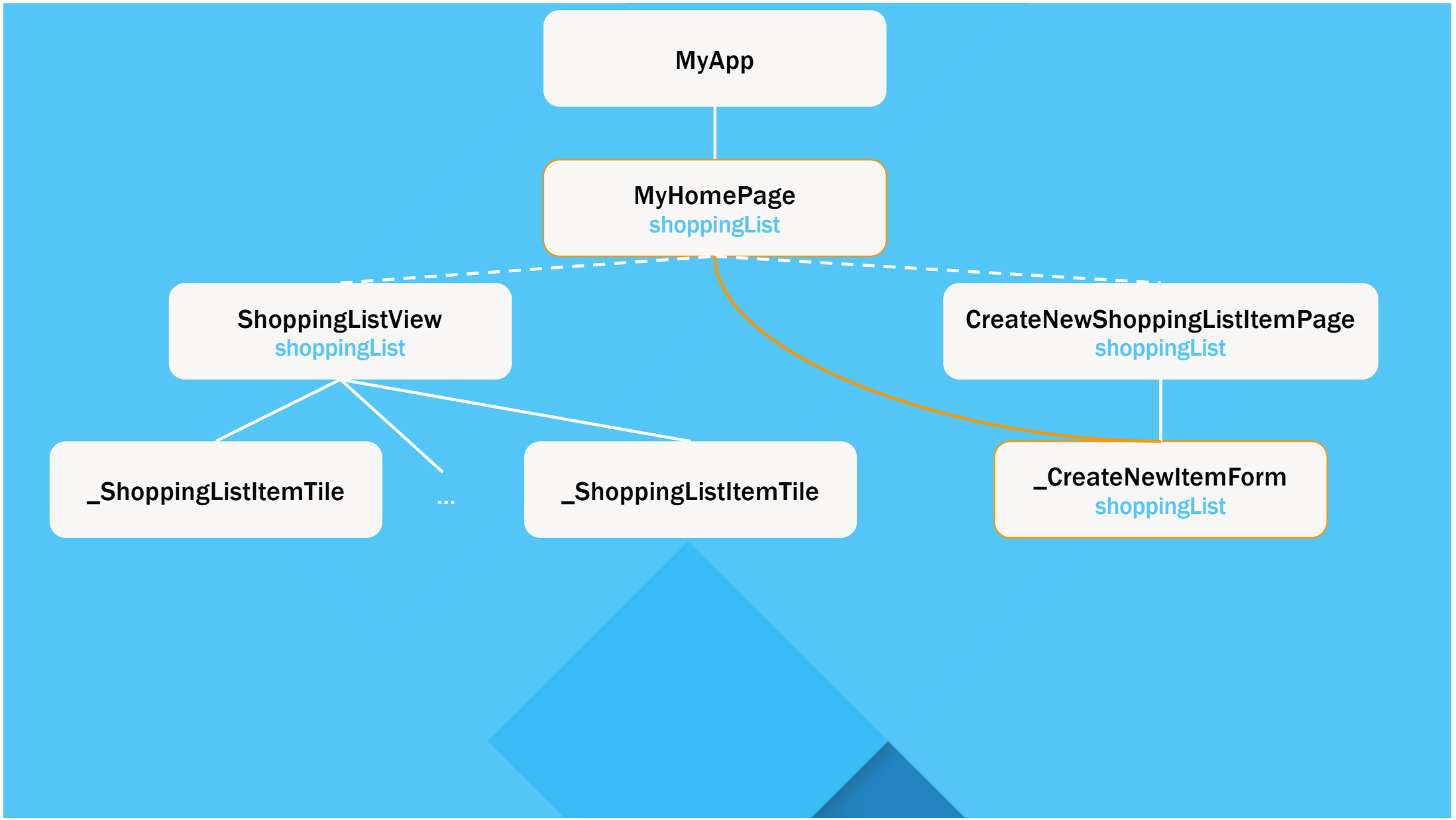

18:02
Create new item
Quantity
Name
Add
18:18
Your shopping list
3x Banana
5x Apple
+
{
sh(
e(
ext) {
eNewShoppingListItemPage(
st: _shoppingList,
ons.add),

Advanced state management

The problem

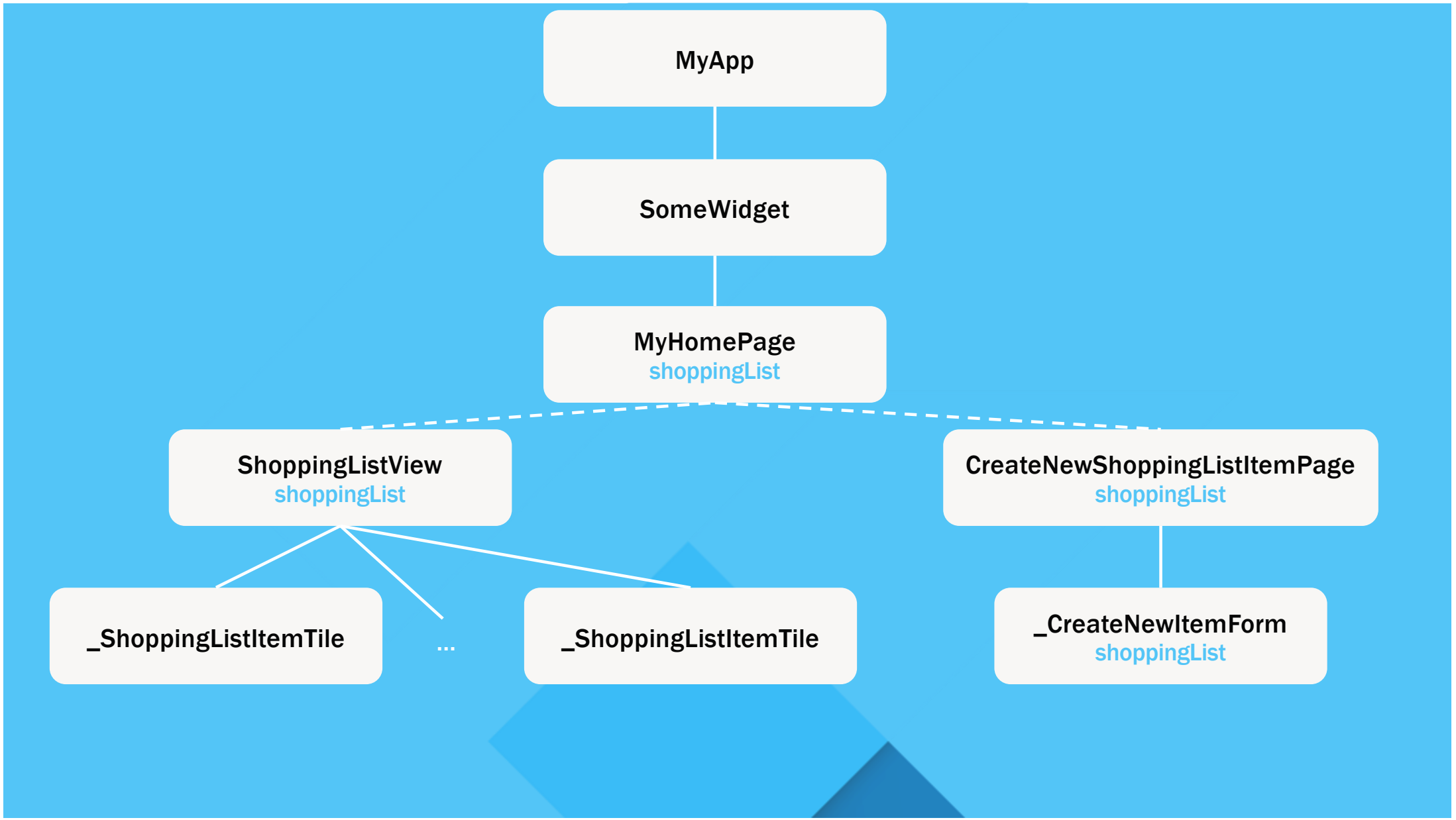

MyApp
MyHomePage
shoppingList
ShoppingListView
CreateNewShoppingListItemPage
_ShoppingListItemTile
...
_ShoppingListItemTile
_CreateNewItemForm

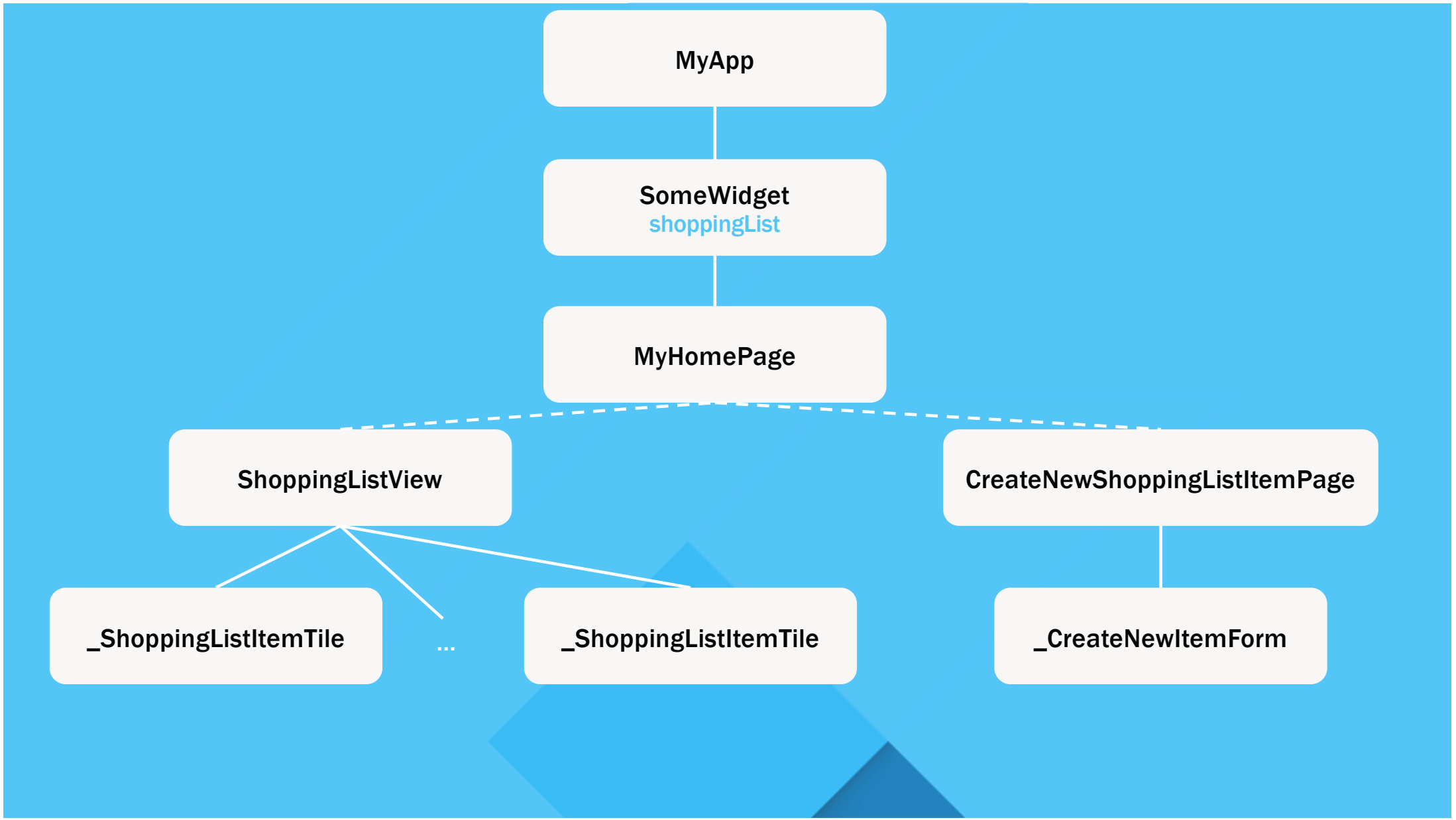

MyApp
MyHomePage
shoppingList
ShoppingListView
shoppingList
CreateNewShoppingListItemPage
shoppingList
_ShoppingListItemTile
...
_ShoppingListItemTile
_CreateNewItemForm
shoppingList

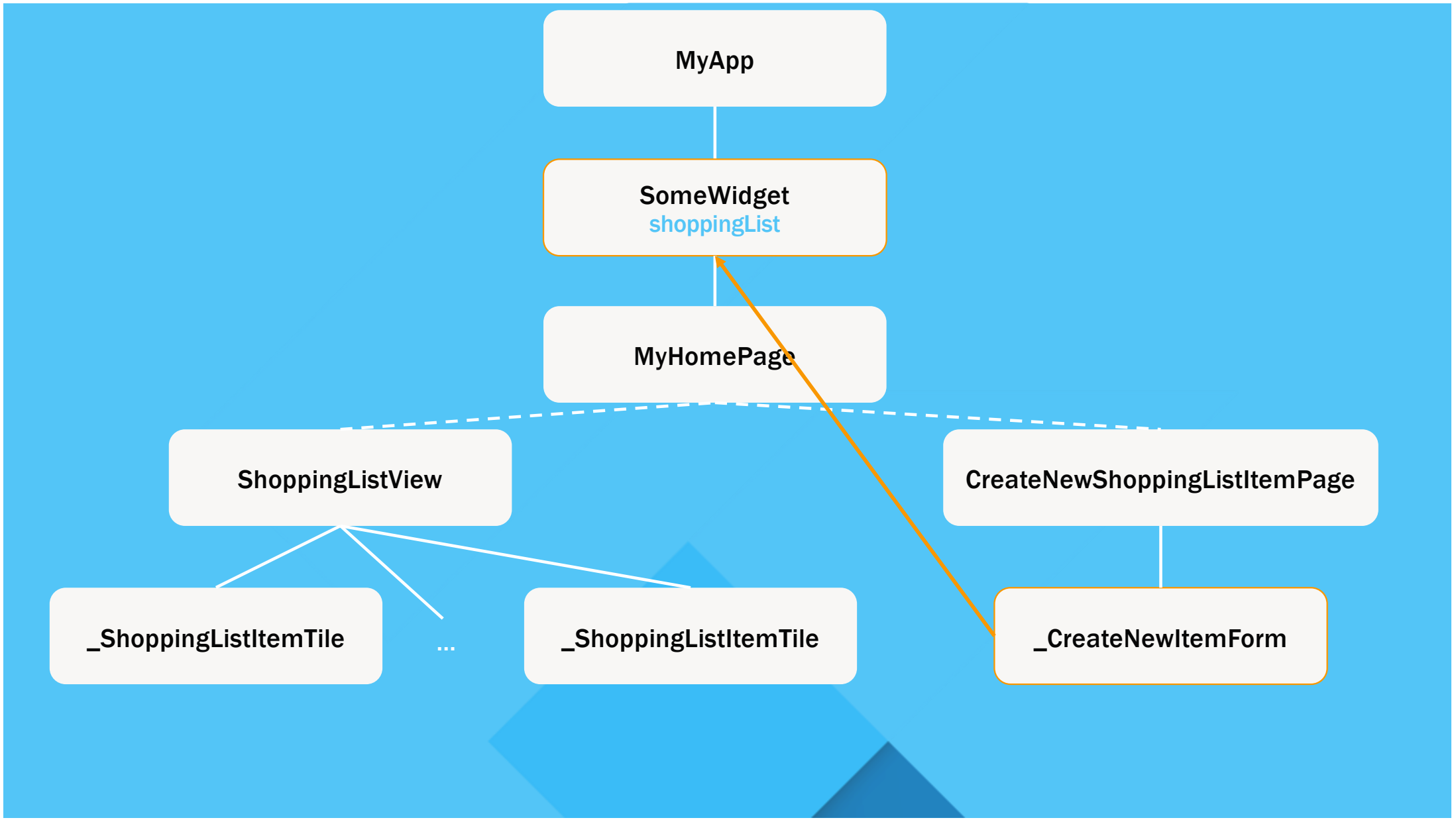

MyApp
MyHomePage
shoppingList
ShoppingListView
shoppingList
CreateNewShoppingListItemPage
shoppingList
_ShoppingListItemTile
...
_ShoppingListItemTile
_CreateNewItemForm
shoppingList

Advanced state management

The idea


MyApp
SomeWidget
MyHomePage
shoppingList
ShoppingListView
shoppingList
CreateNewShoppingListItemPage
shoppingList
_ShoppingListItemTile
...
_ShoppingListItemTile
_CreateNewItemForm
shoppingList


MyApp
SomeWidget
shoppingList
MyHomePage
ShoppingListView
CreateNewShoppingListItemPage
_ShoppingListItemTile
...
_ShoppingListItemTile
_CreateNewItemForm


MyApp
SomeWidget
shoppingList
MyHomePage
ShoppingListView
CreateNewShoppingListItemPage
_ShoppingListItemTile
...
_ShoppingListItemTile
_CreateNewItemForm

Advanced state management

Options

Provider

bloc

riverpod

redux

GetX

New Flutter Devs:

Can you tell what's the best state management option for flutter ?

Flutter Team :

Too many options

Summary

- Widgets are used to configure other widgets
- There is a widget for everything
- Rerender using setState
- Navigate between pages using the Navigator API
- Use lifecycle methods to handle lifetime of created objects

Practice makes perfect

Build your own app

Requirements

- Flutter installation
- A physical device or an emulator
- A little patience
- Creativity

Assignment

- Brainstorm: Take 5 minutes to conceptualize a simple app idea
- Initialize: Create a new Flutter project
- Design: Use your gained knowledge about widgets (Scaffold, AppBar, Text, ...) to build the UI
- Function: Implement basic functionality and use your knowledge on state management
- Style: Add a touch of customization to make your app stand out

It’s a competition

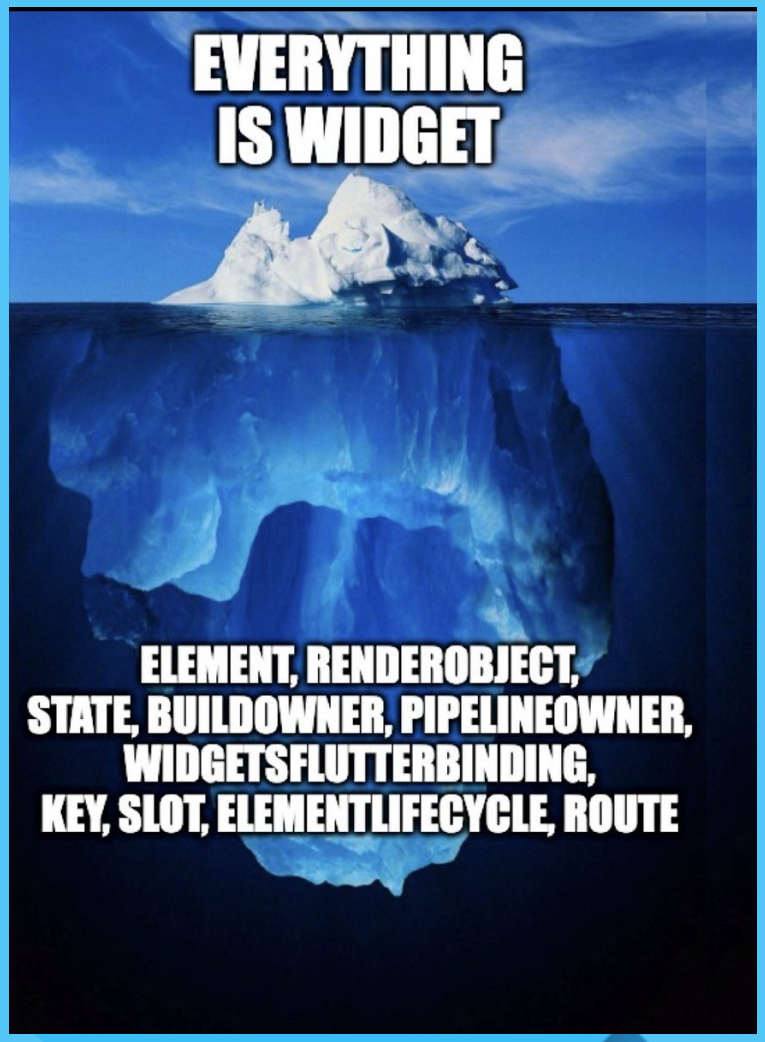
Don’t be afraid to ask for help

Apps & slides

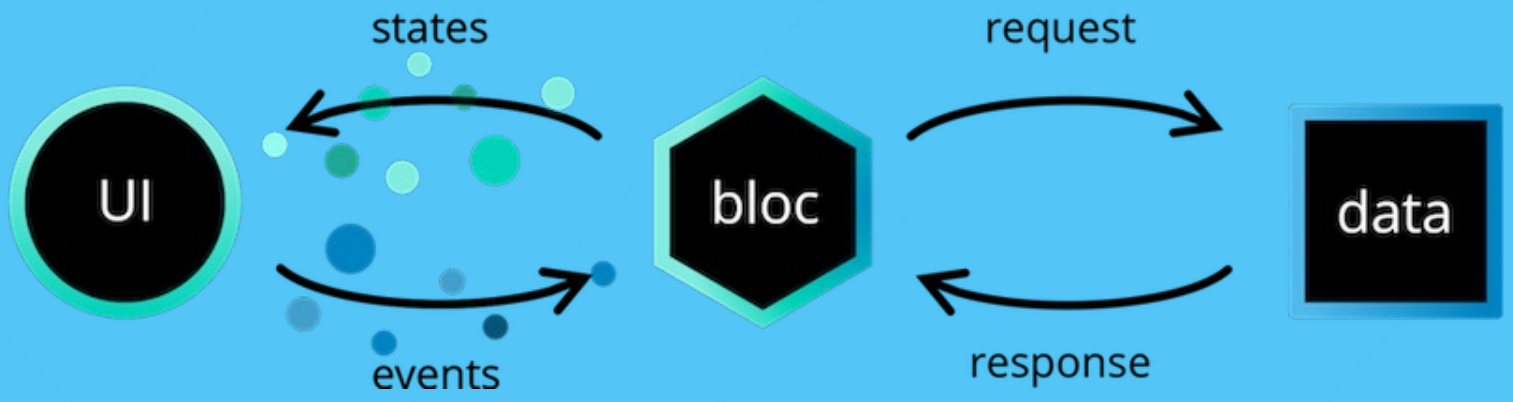

THE WINNER IS
...
makeameme.org


WELL LOOKS LIKE WE
HAVE A WINNER
memegenerator.net

Outlook


EVERYTHING
IS WIDGET
ELEMENT, RENDEROBJECT,
STATE, BUILDOWNER, PIPELINEOWNER,
WIDGETSFLUTTERBINDING,
KEY, SLOT, ELEMENTLIFECYCLE, ROUTE


Flutter Favorite
Flutter Favorite


states
request
UI
bloc
data
events
response

Things to lookup yourself

- InheritedWidget
- Element & RenderObject
- Bloc & Provider
- Asynchronous Dart => Call a REST-API from your Flutter app
- Freezed & JSON Serializable
- Widget/Unit/Integration Tests
- Local Storage